[redacted]

Il6r
no.500
UIUCDCS-R-72-500
cop.2

Math

QAS Question-Answering System

by

Marion Arthur Pumfrey

February, 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-72-500

QAS QUESTION-ANSWERING SYSTEM*

by

Marion Arthur Pumfrey

February 1972

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*This work was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, February 1972.

ACKNOWLEDGMENT

I wish to thank my advisor, Professor Sylvian R. Ray, for his guidance throughout the preparation of this thesis.

I especially wish to thank my wife, Diana, and daughters, Laura and Kimberly, for another year and a half leave of absence.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This study deals with the field of question-answering, QA, systems. I undertook the study to gain insight into the difficulties involved in creating and utilizing a QA system and into QA system capabilities once developed.

In particular, the study deals with QA systems applied to the field of current political intelligence. I desired to answer the following questions: 1) Can machines or relatively untrained clerks easily encode unstructured intelligence information of a wide variety (political, social, economic, industrial, military, target, weather, terrain, weapons capability, etc.) into a form which can be processed by computer? 2) Can general computer QA routines process a wide variety of intelligence information and store intelligence data in a consistent memory structure? and 3) Can computer inference or deductive capabilities be developed to the point where machines assist the intelligence analyst significantly?

The trememdous volume of intelligence information collected today by various means (electronic, photographic, human sources, etc.), the extremely difficult task of properly screening, evaluating, and analyzing this quantity of information, and the difficulty of estimating the intentions or capabilities of another power makes an affirmative answer to these three questions highly desirable.

Of course, this study's scope is far too restricted to fully realize the three stated objectives. However, the study does give the desired insight into QA system design difficulties, utilization problems, and capabilities.

The question answering system, called QAS, developed on the following pages is based on a semantic network and deals with political intelligence which is processed into a somewhat restrictive set of relations. Non-semantic network QA techniques are not discussed. Likewise, machine processing of natural language into a set of relations is extremely complex and is not discussed in this paper. However, trained intelligence clerks could easily and rapidly encode relatively general intelligence information into these relations. These relations convey natural language concepts or meanings; but the relations are easier to process by computer than the corresponding natural language. Further computer processing involves: 1) determining if the information is already represented or partially represented in the memory network, 2) adding the information to the memory network in a consistent manner when the information is not already represented, and 3) answering questions based on the information represented in the memory network.

I develop an algorithm which accomplishes 1) and 2) and answers questions which have their answers "explicitly" represented in the memory network. In addition, QAS furnishes a great deal of immediately related information which is available in the memory network with the answer to each question. However, QAS does not have a strong inferential or deductive capability at this time.

2. DATA BASE MEMORY REPRESENTATION

QAS inputs intelligence information as a set of relations between subjects and objects. QAS stores the set of relations in memory as a complex network of nodes (subjects and objects) and pointers (to subject or object nodes, to property lists, to dictionary entries, and to subnodes or supernodes). The following paragraphs describe the representation of intelligence data in this memory network.

2.1 Nodes

Each node represents the meaning of a word, sentence, paragraph, or concept. There are two types of nodes--basic and complex. Basic nodes have no subset or relational pointers. The time node is one possible basic node. The time node specifies a time period and has the form (T_1-T_2) where T_1 or T_2 can be Unk (unknown) and T_2 can be P (present) which indicates that the time period extends up to the present. If not Unk or P, T_1 and T_2 have the form (year, month, day, time) where the last three items are omitted when unknown. The time is expressed in GMT and is based on the 24 hour clock. For example, 197108171930 is 1930 hours GMT, 17 August 1971. The circled nodes in Figures 1, 2, 3, and 4 are examples of basic nodes.

Complex nodes consist of a subset pointer, a set (possibly empty) of superset pointers, and a set of relational pointers to object nodes which modify or define the concept expressed in the complex node. When QAS stores information, it may find that a concept is already partially represented in the memory network. To fully represent the concept, QAS must add one or more relational pointer(s). When concept intersections of this nature occur, QAS 1) establishes a new node, 2) creates a subset pointer from the new node to the already represented concept, 3) adds a superset from the already stored concept to the new node, and 4) adds the required relational pointers to the modifying concepts. If the relational pointer is a symmetric relation or an active verbal relation, then QAS 1) establishes a second new node, 2) creates a subset pointer to the already stored modifying concept from the second new node, 3) adds a superset pointer from the already stored modifying concept to the second new node, 4) adds the converse relation or symmetric relation from the second new node to the first new node, and 5) adds the required relational pointer from the first new node to the second new node rather than to the already stored modifying concept. For example, QAS represents "The formal governmental structure of North Vietnam (NVN) is distinct from that of the ruling Communist party." as shown in Figure 1. When QAS adds "The ruling Communist party is known as the Lao Dong Party." to the data base, we obtain the configuration of Figure 2. Node(P3) has a subset pointer to the intersecting concept "ruling Communist party" and the relational pointer, Equivalent, to Node(P5) which was created for the symmetric Equivalence relation to Node(P3). Node(P5) has a subset pointer to Node(P4) which was created by the first procedure described above. The nodes in Figures 1 and 2 are depicted as words but actually consist of dictionary, subset, superset, and relational pointers in the memory network. Nil indicates the absence of subset and superset pointers. See Section 3 for a detailed explanation.

Likewise, QAS may find that a concept is already represented in memory which has more relational pointers than the condept it is processing. QAS then 1) constructs a new node with the set of relational pointers required to represent the intersection of the two concepts, 2) erases the duplicated relational pointers from the stored concept node and the subset pointer from the stored concept to its subnode, 3) erases the superset pointer from the subnode of the stored concept to the stored concept node, 4) adds subset pointers from the new node to the original subnode of the stored concept and from the stored concept node to the new node, and 5) adds superset pointers from the subnode of the stored concept to the new node and from the new node to the stored concept node. For example, "the ruling Communist party" is represented in memory as shown in Figure 3. If QAS adds "The Communist party is equivalent to the Lao Dong Party." to the memory network then we have the memory configuration of Figure 4. First, QAS constructs Node(P2) to represent the intersection of the two concept--"communist party". Second, QAS erases the subset pointer from Node(P1) to Node (P) and the Mod relational pointer from Node(P1) to Node(C). Next, QAS erases the superset pointer from Node(P1) to Node(P2). Next, QAS adds subset pointers from Node(P1) to Node(P2) and from

Node(P2) to Node(P). Finally, QAS adds superset pointers from Node(P) to Node(P2) and from Node(P2) to Node(P1). See Section 3 for a detailed explanation.

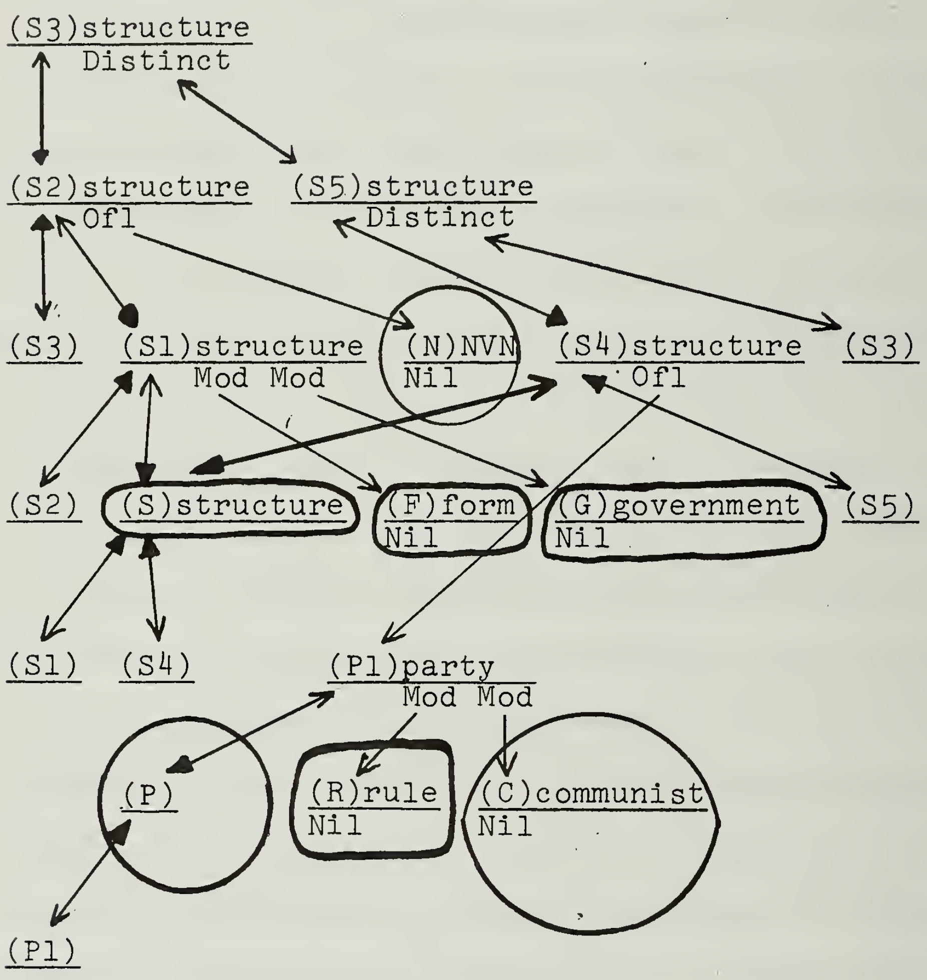


Figure 1. Memory representation of "The formal governmental structure of NVN is distinct from that of the ruling Communist party."

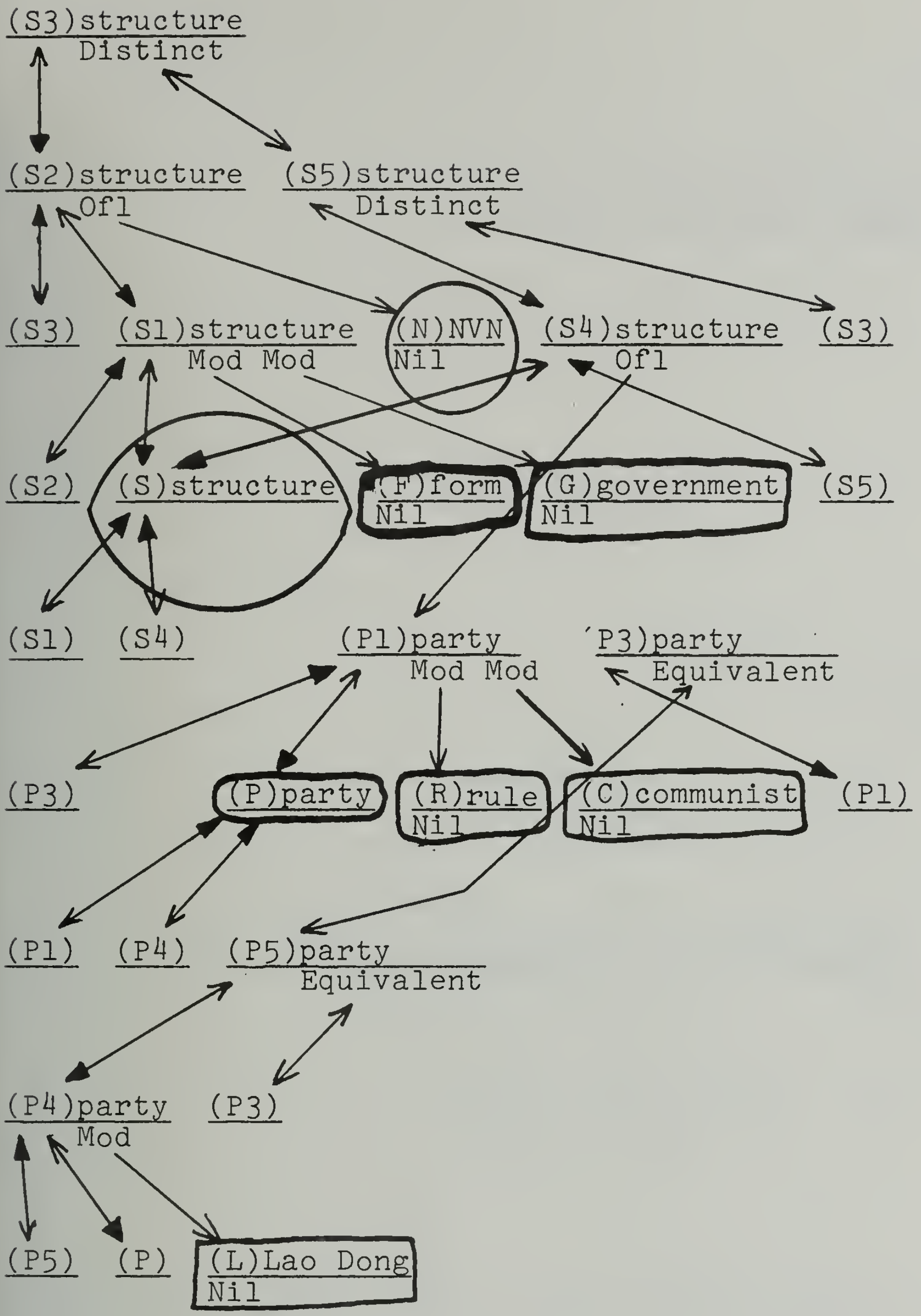


Figure 2. Memory representation of "The formal governmental structure of NVN is distinct from that of the ruling Communist party." and "The ruling Communist party is known as the Lao Dong Party."

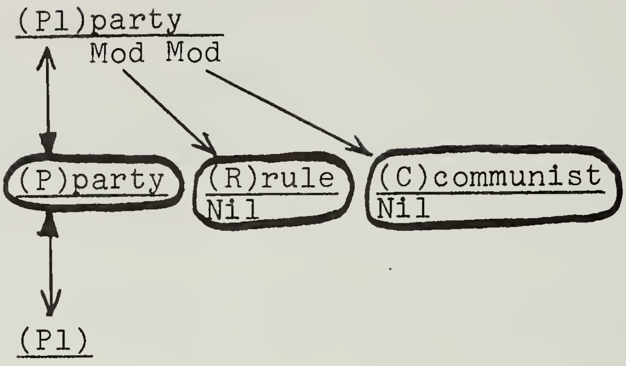


Figure 3. Memory representation of "the ruling Communist party"

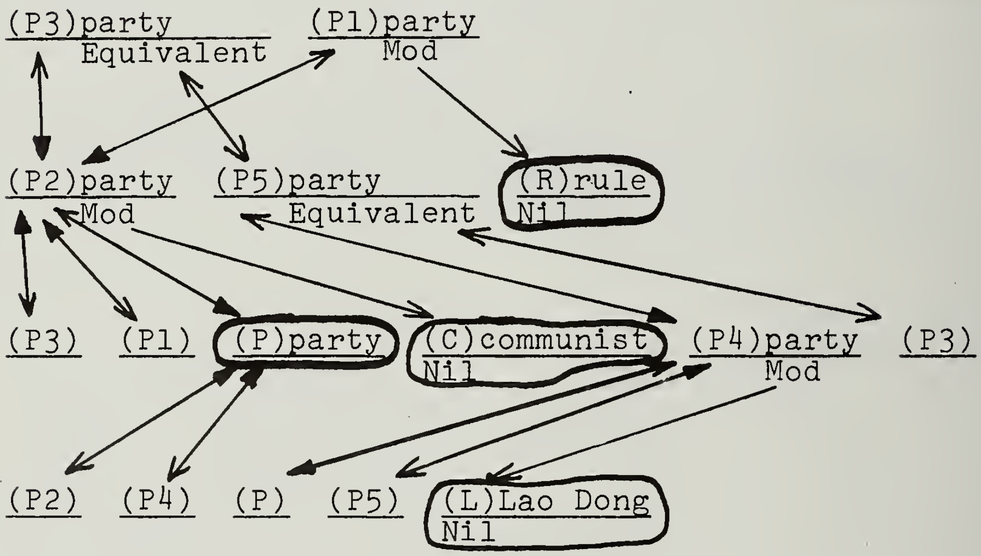


Figure 4. Memory representation of "the ruling Communist party" and "The Communist party is equivalent to the Lao Dong Party."

2.2 Relations

Originally QAS utilized three types of relations--Unary, Binary, and Relational. Country(x), where x is the name of a country, is an example of a unary relation. Relations 1-7 of Appendix A are the unary operations originally used in QAS. They are unnecessary and have been dropped from the system.

There are three types of binary relations--Modifier, Normal, and Connective. Mod(x,y) is the only Modifier relation and means that x is modified by y in the adjectival sense.

Relations 9-22 of Appendix A are elements of one subset of the set of all possible Normal relations. A subset of these relations was used to encode the text of Appendix B. The following are examples of Normal relations: Of1(x,y), Equivalent(x,y), and Define(x,y). Of1 is used as a function word to indicate belonging or the possessive relationship. Equivalent means corresponding or virtually identical especially in effect or function. Define means to make distinct in outline. The three examples demonstrate how prepositions, set relationships, and verbs are utilized as Normal relations. Each relation obtained from an active verb has a corresponding converse relation. For example, Define(x,y), x defines y, has the converse relation Define-1(y,x), y is defined by x. Likewise, each symmetric relation, R(x,y), has a corresponding relation R(y,x). The negation of Normal relations is also possible and, for example is written as N-equivalent(x,y), x is not Equivalent to y.

The set of Normal relations is large. However, the number of such relations can probably be held to a workable number when encoding information in any one field of interest, such as political intelligence, and when the encoder uses one relation where several relations serve equivalent or analogous functions. For example, Equivalent(x,y) and Equal(x,y) are identical under many interpretations. In addition, reduction possibilities, such as Distinct(x,y) = N-equivalent(x,y) should be exploited fully.

And(x_1,x_2) is an example of a Connective relation. The encoder utilizes Connective relations when the original text contains a single relation which acts on more than one subject. Connective relations require that QAS establish two relational pointers--one for each subject or object joined by the Connective relation. Relations 23 and 24 of Appendix A are examples of Connective relations. However, QAS allows only the "And" connective at this time.

Relational relations have numerous possible forms. Essentially, relational relations capitalize on the symmetry or transitivity of relations when it exists. For example, Equivalent(x,y) implies Equivalent(y,x) and Ofl(x,y) and Ofl(y,z) implies Ofl(x,z). Negation is another source of relational relations. Equivalent(x,y) = N-distinct(x,y) is one possible example. In addition, less obvious relational relations frequently become apparent while attempting to answer questions in QAS. For example, the object of the preposition "of" used in the possessive sense is often used as an adjective with the same meaning. For instance, "the governmental system of North Vietnam (NVN)" can be written equivalently as "the system of government of NVN". Hence, Mod(x,y) = Of1(x,y) is a possible relational relation in some contexts. Relations 25-27 of Appendix A are examples of relational relations.

Relational relations are used in the question-answering phases of QAS. The answer to a question is the goal of the QA routines. When the goal cannot be directly satisfied all possible equivalent goals or subgoals are considered before QAS states that the answer is unknown. These equivalent goals or subgoals can be obtained in part from the relational relations.

2.3 Linkage

QAS utilizes two primary types of pointers to establish its information network--internode and dictionary-to-memory or memory-to-dictionary pointers. QAS establishes at least one internode pointer for each relation in the input code unless QAS determines that the same subject, relation, and object are already represented in the memory network. See Paragraph 2.4 for a description of the input code. Each pointer is tagged to identify the type of relation it represents. QAS creates a single pointer for Mod(x,y) and for prepositionally derived relations. The pointer is from the subject x to the object or modifier y. QAS establishes two pointers for each symmetric relation--one from the relation's subject to its object and a second from the relation's object to its subject. Likewise, QAS establishes two pointers for each relation derived from an active verb. One pointer, from the relation's subject to its object, represents the relation itself. A second pointer, from the relation's object to its subject, represents the relation's converse. In addition, a third pointer is associated with each of the latter two pointers for active verbal relations. This third pointer points to a property list which consists of four pointers--place, time, agency, purpose. These four pointers point to nodes which represent the following concepts respectively: 1) geographic location in which the event occurred, 2) time period over which the action was effected, 3) the method used to achieve the relation's result, and 4) the purpose for which the action was taken. In addition, these four pointers may be marked Nil (null), Unk (unknown), or NA (not applicable) if the required concept is not used, unknown to the coder, or not applicable to this relation. In addition, the subset-superset pointers disucssed in paragraph 2.1 are present.

For example, "The formal governmental structure of NVN is defined in the 1960 Constitution." has the memory representation shown in Figure 5. The Mod relation is represented by a single pointer from the subject to the node which modifies the subject. The concept "structure", Node(S1), is modified by the concepts "formal" and "government", Node(F) and Node(G) respectively. A single pointer from Node(S2) to Node(N) represents the prepositional relation OF1. The relation Define stems from an active verb and is therefore represented by two pointers in memory. The filled in arrow from Node(C2) to Node(S3) depicts the relation itself while the open arrow from Node(S3) to Node(C2) represents the converse relation. The horizontal arrow depicts the property list pointer attached to the relation Define. The property list is enclosed in parenthesis and contains pointers to the place concept, NVN - Node(N), and the time concept, 1960 - present, Node(M). The agent and purpose pointers are marked unknown. In Figure 6, the two pointers are marked unknown. In Figure 6, the two pointers which represent the symmetric relation, Distinct, are depicted by a double ended arrow between Node(S2) and Node(S4). Double ended arrows represent subset-superset pointers as shown in Figures 1-10. The filled-in arrow represents the subset pointer, while the open arrow represents the superset pointer.

Each word concept in memory has a pointer to it dictionary definition. Each word meaning in the dictionary has a pointer to all the nodes utilizing a specific meaning of the word to represent a concept in memory. For example, Figure 6 shows the dictionary-to-memory and memory-to-dictionary pointers for "The governmental structure is distinct from the structure of the Communist party."

2.4 Encoding Intelligence Into QAS Relations

I will describe the encoding procedure by working an example. The first sentence of the text is: "The formal governmental structure, as defined in the 1960 Constitution, is distinct from that of the ruling Communist party, known as the Lao Dong

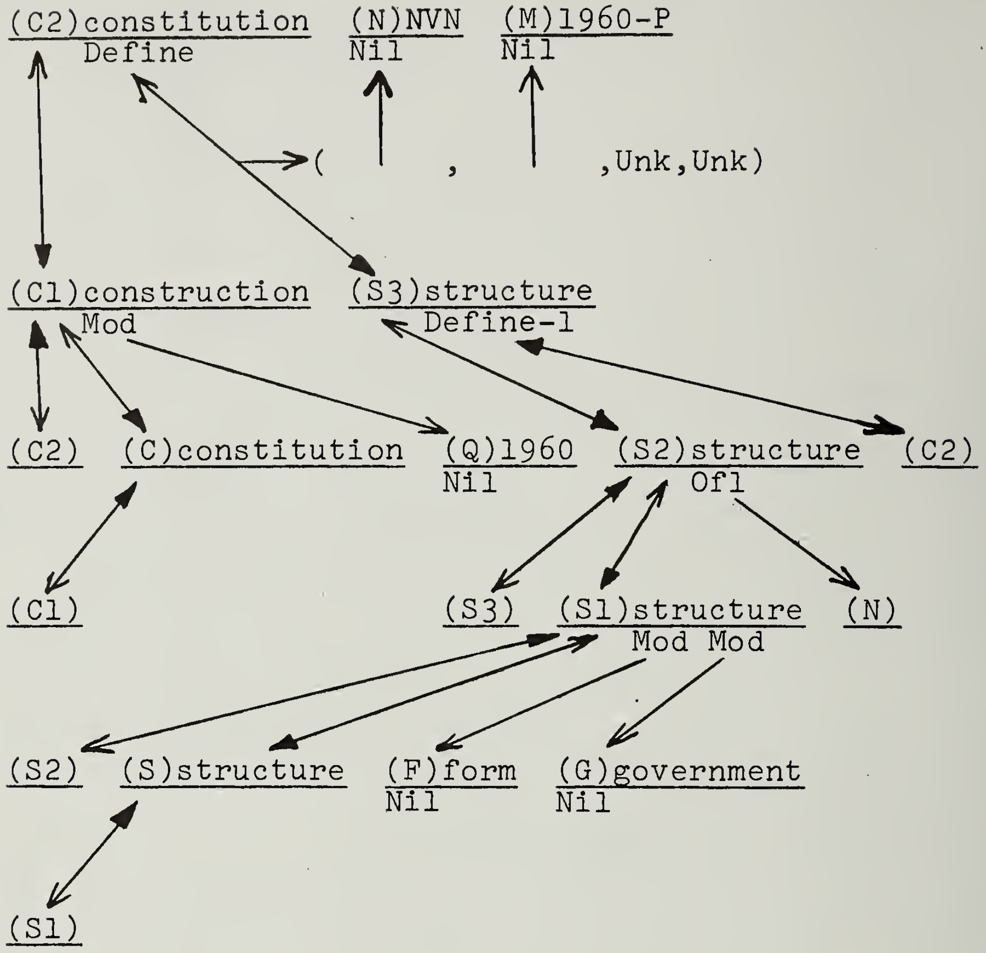
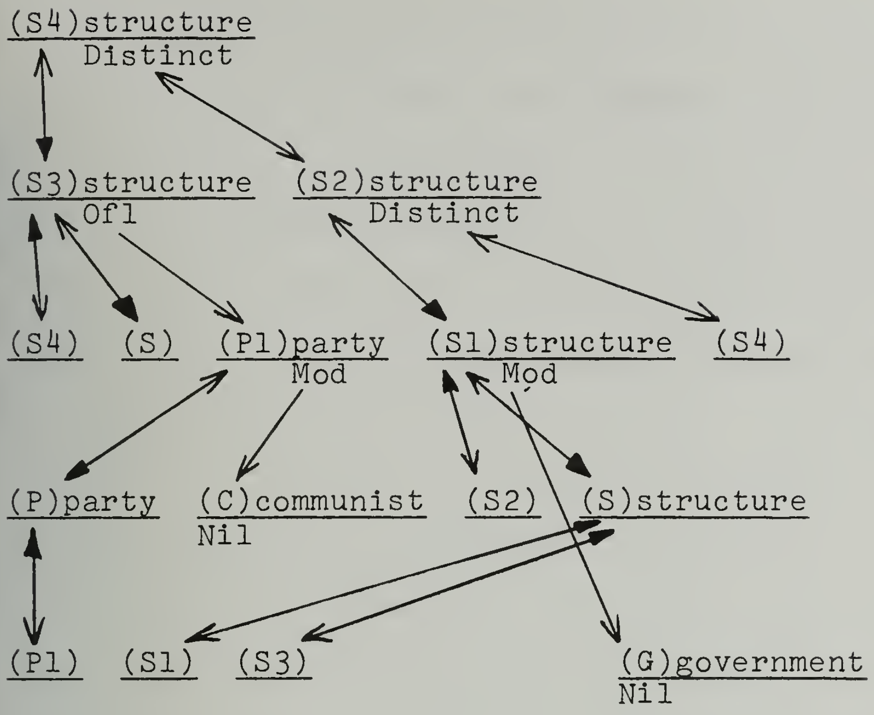


Figure 5. Memory representation of "The formal governmental structure of NVN is defined in the 1960 Constitution."

Party, which is not mentioned specifically in the Constitution as an integral component of the governmental system." First, the encoder breaks this complex sentence down into several simple sentences which convey approximately the same meaning as the original sentence. In addition, the encoder adds known quantifiers such as name, place, and time during the subsentence construction process. The addition of these quantifiers facilitates the search for identical concepts when adding data to the memory network as described in Section 3.



Dictionary

structure (S, S1, S2, S3, S4)

government (G)

party (P, P1)

communist (C)

Figure 6. Memory representation of "The governmental structure is distinct from the structure of the Communist party", and pointers to and from the QAS dictionary

For example, the encoder can re-write the original text sentence as the following subsentences which have the known quantifiers added. 1) The 1960 Constitution of NVN defines the formal governmental structure of NVN. 2) The formal governmental structure of NVN is distinct from the structure of the ruling Communist party of NVN. 3) The ruling Communist party of NVN is equivalent to the Lao Dong Party of NVN. and 4) The Lao Dong Party of NVN is not an integral component of the governmental system of NVN.

The encoder codes the first subsentence as: Define/NVN, 1960-P,Unk,Unk/(Ofl(Mod(constitution,1960),NVN),Ofl(Mod(structure,And(form,government)),,NVN)). Define is an active verb and therefore requires the property list, /place, time, agency, purpose/, which is enclosed in slashes.

Utilizing the relations defined in Appendix A, the encoder codes the latter three subsentences as: 2) N-equivalent (Ofl(Mod(structure,And(form,government)),NVN),Ofl(structure,Ofl (Mod(party,And(rule,communist)),NVN))), 3) Equivalent(Ofl(Mod (party,And(rule,communist)),NVN),Ofl(Mod(party,Lao Dong),NVN)), and 4) N-component(Ofl(Mod(party,Lao Dong),NVN),Ofl(Mod(system, government),NVN)).

N-equivalent and Component are not active verbs and therefore do not require a property list. Sentence four is negated in agreement with the text's meaning. As yet, QAS does not provide a means to represent the adverbial modifier, integral, on the relation, Component. Appendix C contains subsentences which have been constructed from the original text and the coded form of these subsentences. Appendix D contains a diagram of the memory network representing the original text.

Trained intelligence clerks can easily encode a wide variety of intelligence information in the QAS format as described above. The set of relations will expand as the intelligence data becomes more diversified. However, the relation set can be held to a manageable size when working with a data base which is oriented to any one type of intelligence information and when the encoder does not expand the relation set when an implemented relation can serve the same function as the newly discovered relation.

3. ADDING ADDITIONAL DATA TO THE MEMORY NETWORK

There are three cases to consider when developing an algorithm for adding additional intelligence data to the QAS memory network: 1) The memory network already exactly represents the new concept. 2) The memory network already represents a portion of the new concept and 3) The new concept is a subset of concepts which are already represented in the memory network. The following paragraphs explain these cases more fully and describe an algorithm which adds additional intelligence data to the memory network.

3.1 Case Considerations

If the concepts expressed in the new intelligence data are the same as concepts already present in the QAS memory network, then QAS performs no further processing. Exactly means that each subject in the input code has no more or no less than the required relational pointer to its object in the memory network.

QAS follows an identity check procedure to determine if a concept is already exactly represented. QAS performs the identity check by 1) finding the basic concepts within the nested concepts of the coded input, 2) looking the subjects of these basic concepts up in the QAS dictionary to find all candidate concepts which have these subjects in the memory network,

3) identifying, if possible, the stored candidate concepts which have the required relational pointers to the basic concepts' objects, and 4) recursively passing the addresses of these identified nodes to the next higher level of the nested input concept to see if the identified nodes have the required relational pointers to the required objects.

For example an identity check on Equivalent(Ofl(counterpart,Ofl(bureau,Ofl(Mod(committee,central),Ofl(Mod(party,Lao Dong),NVN)))),Inl(And(ministry,commission),Ofl(government,NVN))) requires that:

1.	counterpart	is related by Equivalent	to ministry
2.	counterpart	Equivalent	commission
3.	counterpart	Ofl	bureau
4.	ministry	Inl	government
5.	commission	Inl	government
6.	committee	Mod	central
7.	committee	Ofl	party
8.	government	Ofl	NVN
9.	party	Mod	Lao Dong
10.	bureau	Ofl	committee

And finally, the identity check on the example above requires that the node representing the objects on the right must be the same as the identically named subjects on the left. For example, the node which represents government in lines four and five and the node which represents government in line eight must be the same.

As described in Section 2.1, case two processing includes: 1) creating a new node with a subset pointer to the node which partially represents the new concept, 2) adding a superset pointer from the partially present concept node to the new node, and 3) adding the relational pointer(s) required to fully represent the new concept from the new node to the required object node(s). Likewise, case three processing includes: 1) creating a new node to replace the identified concept intersecting node, 2) adding the required intersecting relation(s) from the new node to the required object node(s), 3) erasing the intersecting relations from the identified concept intersecting node and replacing its subset pointer by a new subset pointer to the newly created node, 4) creating a second new node with a subset pointer to the first new node if unrepresented relations exist, 5) adding the new concept's unrepresented relation(s), if any, from the second new node to the required object(s), 6) replacing the superset pointer from the subnode of the identified concept intersecting node to the concept intersecting node with a superset pointer to the first new node, and 7) adding superset pointers from the first new node to the concept intersecting node and from the first new node to the second new node.

QAS utilizes the identity check procedure to determine if the memory network already partially represents the new concept or if the new concept intersects concepts already represented in the memory network. If the required relation is symmetric or an active verbial relation, then QAS follows the identity check procedure to determine if the identified object node has the required symmetric or converse relation to the identified subject node.

For example, if the memory network contains the concepts represented in Figure 7, then QAS takes the following actions while adding Equivalent(Ofl(government,NVN),Mod(government,communist)) to the memory network. Figure 8 depicts the resultant memory representation. QAS eliminates Node(1) and Node(2) because they fail the first identity check requirements. QAS, therefore, constructs a new node, Node(6), adds a subset pointer from Node(6) to Node(3), and adds a superset pointer from Node(3) to Node(6). Since Node(2) partially satisfies the second identity check, QAS creates Node(5), the required subset-superset pointers between Node(2) and Node(5), and the Mod relational pointer to Node(7) to represent "communist government. Next, QAS creates Node(4) and the required subset-superset pointers between Node(4) and Node(5). Finally, QAS adds the symmetric relations, Equivalent, between Node(4) and Node(6).

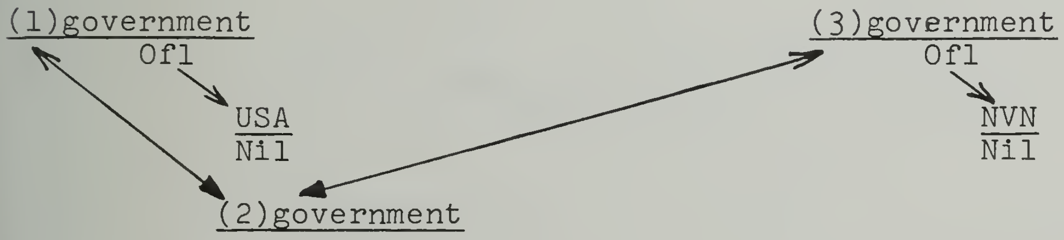


Figure 7. Memory representation of three governmental concepts

If the memory network contains the concepts represented in Figure 9, then QAS takes the following actions while adding Equivalent(Ofl(Mod(party,communist),NVN),Mod(party,Lao Dong) to the memory network. Figure 10 depicts the resultant memory representation. QAS 1) determines that "communist party of NVN is already present in the memory network, 2) creates Node(10) and the required subset-superset pointers between Node(10) and Node(3), 3) creates Node(12) to represent "Lao Dong", 4) creates Node(11), the subset-superset pointers between Node(11) and Node(0), and the Mod Relational pointer from Node(11) to Node(12), 5) creates Node(13) and the required subset-superset pointers between Node(13) and Node(11), and 6) finally establishes the Equivalent relational pointers between Node(13) and Node(10).

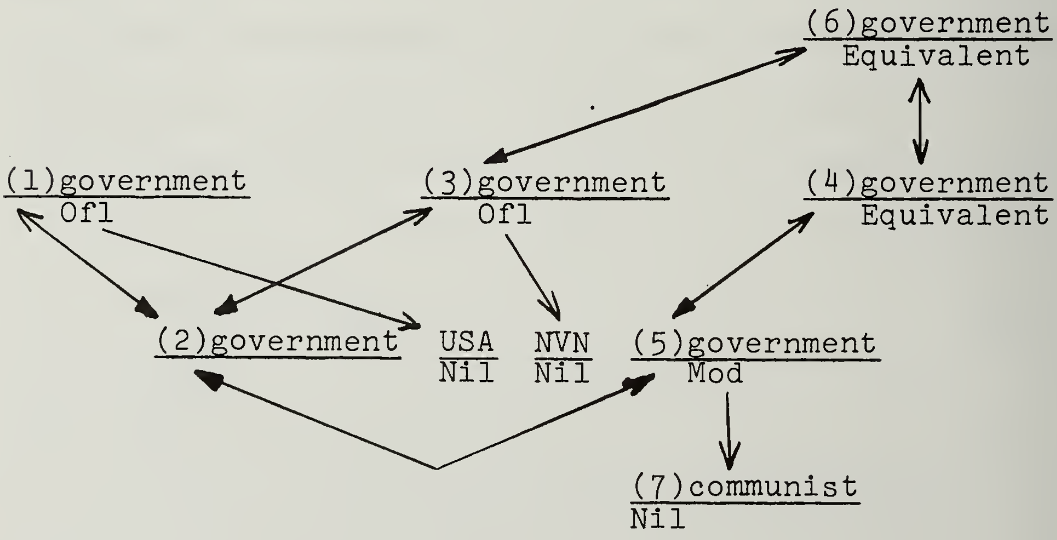


Figure 8. Memory representation of six governmental concepts

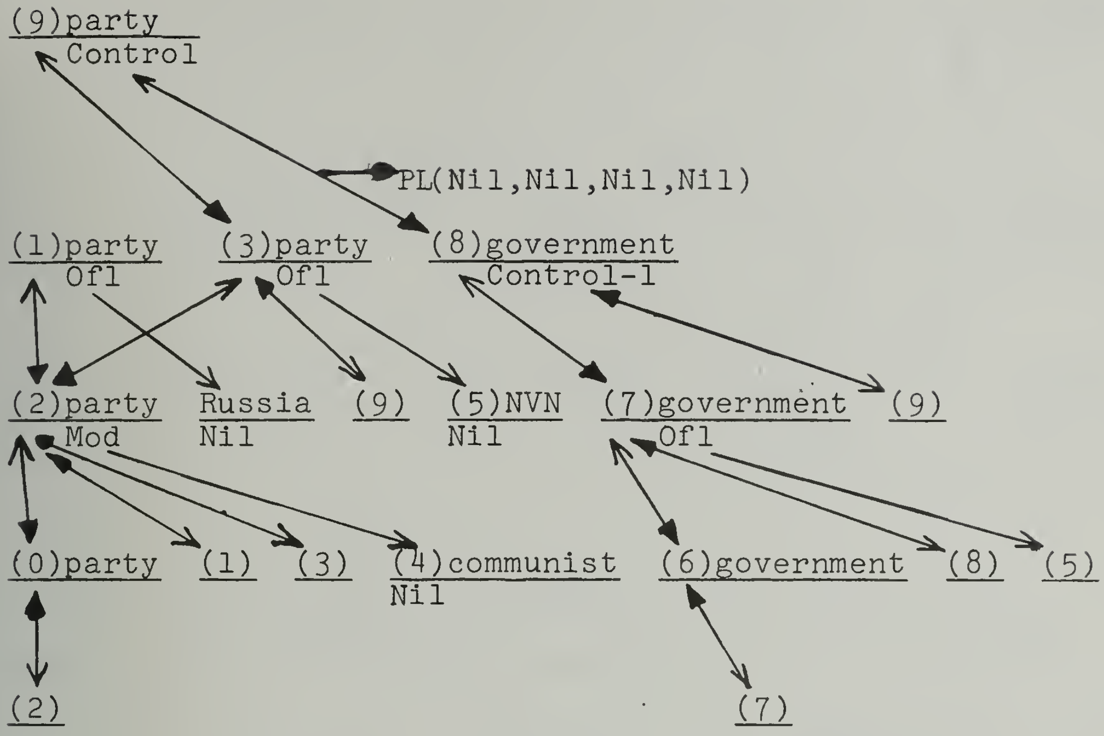


Figure 9. Memory representation of three concepts involving government and five concepts involving party

3.2 Algorithm

The following algorithm uses four parallel stacks which contain an address, relation, subject, and object respectively. The address is S (subject stack), or O (object stack), or F (terminal condition). The techniques used to accomplish each step (if unclear) is explained more fully in the paragraphs following the algorithm.

Step I (Initialize)

A. Set address = F

B. Set string = encoded concepts

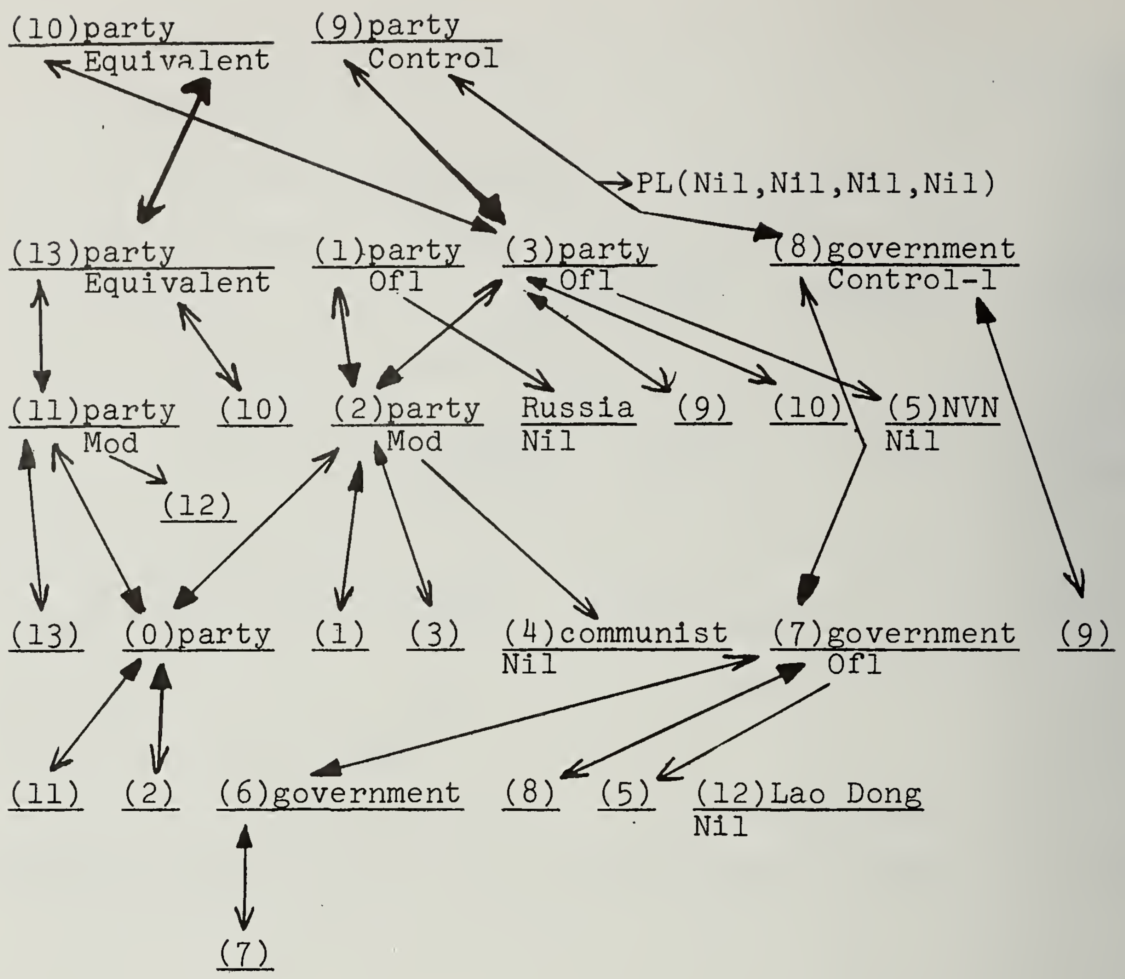


Figure 10. Memory representation of eight concepts of party and three concepts of government

Step II (Break up string and push onto stacks)

A. Break string into relation, subject, and object

B. Push address, relation, subject, and object onto the four parallel stacks

Step III (Check for simple subject and object) (A simple subject or object is a subject or object which does not have a relation prefix.)

```
A. If top of subject stack ≠ simple subject
    1. Set string = (complex) subject
    2. Set top of subject stack = null
    3. Set address = S
    4. Go to Step II
B. If top of object stack ≠ simple object
    1. Set string = (complex) object
    2. Set top of object stack = null
    3. Set address = 0
    4. Go to Step II
```

<u>Step IV</u> (Set up candidate list of required) (A candidate list is a list of pointers. QAS looks the subject or object up in the dictionary and copies all dictionary pointers to concepts utilizing this word into the candidate list. Hence, a candidate list is an n-tuple, (pointer 1,...,pointer n). The candidate list tends to be ordered from the simplest concepts to the most complex concepts. QAS must create the candidate list at this point rather than in Step I. New nodes are frequently added to the memory network in the remainder of the algorithm. Hence, any candidate list created in Step I would probably not contain all required candidates when the candidate list was finally manipulated by the algorithm. If QAS does not find a word during dictionary lookup, it adds the word to the dictionary and creates a node without any relational or subset-superset pointers to represent that word. A *subject or *object node represents lower level concepts that have already been processed by this algorithm.

```
        A.  If top of subject stack ≠ to *subject
            1.  Replace subject with its candidate list
        B.  If top of object stack ≠ to *object
            1.  Replace object with its candidate list
```

Step V (Form "and list" if required) (The "and list" is a list of concepts enclosed in +'s. *pointer is normally the address of the concept node located--if the concept being processed is already in memory--or created by the identity check routine.)

```
        A.  If relation = And
            1.  Set *pointer = *+ subject object +
            2.  Go to Step VII
```

Step VI (Identity Check)

A. Perform subject, relation, object identity check and create required concept node if concept is not already stored in memory.

Step VII (Store *pointer, pop stacks, and check for task completion)

```
        A.  If top level of address stack = F
            1.  Stop
```

B. Store *pointer in the next lower stack level and at the address specified in the top level of the address stack

```
        C.  Pop all four parallel stacks
        D.  Go to Step III
```

QAS must break each complex string into relation, subject, and object. The break up procedure is: 1) Scan right to the first "(". The word accumulated at this point is the relation.

2) Scan past the first "(". 3) Scan right to the first "," which is preceded by a character string balanced with respect to parenthesis. The balanced character string is the subject. 4) Scan past the first ",". 5) Scan right to the first ")" which is preceded by a character string balanced with respect to parenthesis. The second balanced character string is the object.

Appendix D depicts the memory network that results after this algorithm acts of the data inputs of Appendix C. Appendix E shows the algorithm's results at each stage of processing the inputs of Appendix C.

3.3 <u>Identity Check</u>

QAS must perform an identity check on each relation, subject, and object in the coded input string. The identity check involves nine cases--a case for each pair in the cartesian product (*subject, candidate list subject, "and list" subject) (*object, candidate list object, "and list" object). In addition, the implementation of converse or object to subject symmetric relations requires special processing. Each of these ten cases are discussed in the following paragraphs.

Case I, *subject and *object. QAS precedes as follows:

```
I. If *subject has the required relation to *object
        A. Set *pointer = *subject
        B. Stop
II. If *subject has superset pointers
        A. If any supernode S has the required relation to
*object
```

```
        1. If supernode S has more than the required relations
            a. Create a new node with the required relational pointer to *object (Creating a new node includes the addition of dictionary to new node and new node to dictionary pointers.)
            b. Erase the required relational pointer to *object and subset pointer to *subject from supernode C
            c. Erase the superset pointer from *subject to supernode C
            d. Add subset pointers from new node to *subject and from supernode C to new node
            e. Add superset pointers from *subject to new node and from new node to supernode C
            f. Set *pointer = new node
            g. Stop
        2. Set *pointer = supernode C
        3. Stop
    B. Create a new node with required relational pointer to *object
    C. Add subset  pointer from new node to *subject
    D. Add superset pointer from *subject to new node
    E. Set *pointer = new node
    F. Stop
III. Go to Step II.B.
```

Case II, *subject and candidate list object. QAS precedes as follows:

I. Set *object = basic candidate node

II. Go to Step I, Case I

Case III, *subject and "and list" object. Case III is actually the same as Cases I and II except that QAS searches for the required relational pointer from *subject to the *object selected for each element of the object "and list" and creates new nodes as described in Case I if the required relation to any of the *objects in the object "and list" is not present.

Case IV, candidate list subject and *object. QAS precedes as follows:

I. Select a candidate subject C to be checked

II. If the candidate node C has the required relational pointer to *object.

A. If subnode of candidate subject C has relational pointers

1. Delete candidate subject C from subject candidate list

2. Go to Step III

B. If candidate node C has more than the required relation to *object

1. Construct a new node with the required relational pointer to *object

2. Erase the required relational pointer from candidate node C

```
        3. Create a superset pointer from the new node to candidate node C
        4. Create a subset pointer from candidate node C to the new node
        5. Erase original subset pointer from candidate node C to its original subnode D
        6. Erase original superset pointer from original subnode D to candidate node C
        7. Create a superset pointer from the original subnode D to the new node
        8. Create a subset pointer from the new node to the original subnode D
        9. Set *pointer = new node
       10. Stop
    C. Set *pointer = candidate node C
    D. Stop
III. If there are more unchecked candidate subjects
    A. Go to Step I
 IV. Create a new node with the required relation to *object
  V. Create a subset pointer from the new node to the basic candidate node
 VI. Create a superset pointer from the basic candidate node to the new node
VII. Set *pointer = new node
VIII. Stop
```

Case V, candidate list subject and candidate list object.

QAS precedes as follows:

I. Set *object = basic candidate node

II. Go to Step I, Case IV

Case VI, candidate list subject and "and list" object. Case VI is actually the same as Cases IV and V except QAS searches for the required relational pointer from the selected candidate node to the *object selected for each element of the object "and list" and creates new nodes as described in Case IV if the required relation to any of the *objects in the object "and list" is not present.

Cases VII, VIII, and IX "and list" subject and *object, candidate list object, or "and list" object. QAS precedes as follows:

I. Set list = null

II. Select an element of "and list" subject

III. Perform *subject or candidate list subject to *object, candidate list object, or "and list" object identity check as required by the subject and object type involved

IV. Set list = list *pointer

V. If there are additional "and list" subject elements which have not been processed

A. Go to Step II

VI. Set *pointer = *+list+

VII. Stop

Case X, symmetric or converse relational pointers. QAS adds relational pointers to concept nodes during the identity check procedures discussed above. The addition of relational pointers require additional processing when the relations are symmetric or are derived from active verbs. Active verb relational pointers require the addition of the converse relation, and symmetric relational pointers require double pointer implementation (from subject to object and from object to subject). To insure that the subset-superset properties of the memory network are not destroyed by the addition of these converse or object to subject symmetric relations, QAS precedes as follows:

I. Create a new node

II. Add a subset pointer from the new node to the required object node

III. Add a superset pointer from the required object node to the new node

IV. Add the required converse or object to subject symmetric relation from new node to the required subject node

V. Add the required relation from the required subject to the new node rather than the normally required object

3.4 Example

Assume the memory network of Figure 11 as a starting point and let the coded input be N-equivalent(Ofl(Mod(structure, And(form,government)),NVN),Ofl(structure,Ofl(Mod(party,And(rule, communist)),NVN))). First, set address = F and string = the coded input above. Next, break string into:

relation = N-equivalent

subject = Ofl(Mod(structure,And(form,government)),NVN)

object = Ofl(structure,Ofl(Mod(party,And(rule,communist)), NVN))

The stacks are:

Level 1	F	N-equivalent	Of1(Mod(structure, And(form,govern- ment)),NVN)	Of1 (structure, Of1(Mod(party,And(rule,commu- nist)),NVN))

The tope of the subject stack has a relation prefix and hence is not simple. Therefore, set string = Of1(Mod(structure,And (form,government)),NVN), top of subject stack = null, and address = S. Break string into:

relation = Of1

subject = Mod(structure,And(form,government))

object = NVN

The stacks are:

Level 2	S	Of1	Mod(structure, And(form,govern- ment))	NVN
Level 1	F	N-equi- valent	----------------	Of1(struc- ture,Of1(Mod(party, And(rule, communist)), NVN))

The top of the subject stack has a relation prefix and hence is not simple. Therefore, set string = Mod(structure,And(form, government)), top of subject stack = null, and address = S.

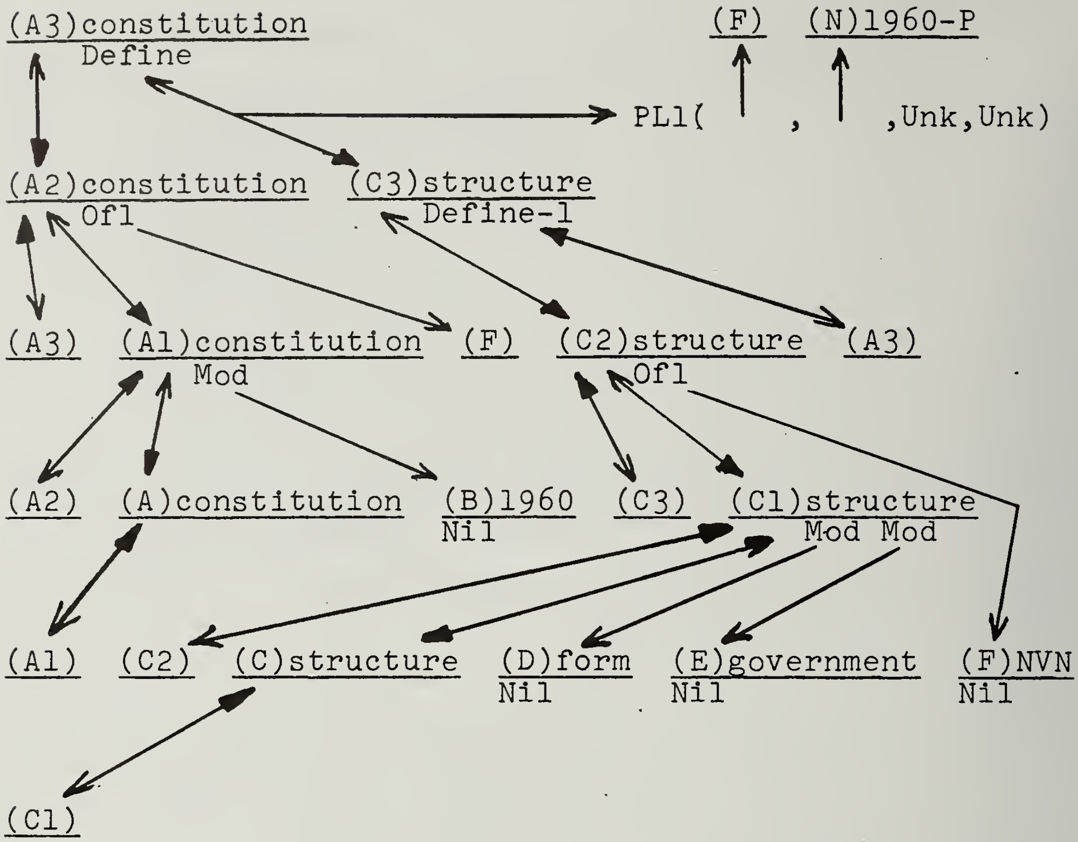


Figure 11. Memory representation of "The 1960 constitution of NVN defines the formal governmental structure of NVN."

Break string into:

relation = Mod

subject = structure

object = And(from, government)

The stacks are:

Level 3	S	Mod	structure	And(form,government)
Level 2	S	Ofl	---------------	NVN
Level 1	F	N-equivalent	---------------	Ofl(structure,Ofl(Mod

(party,And(
rule,commu-
nist)),NVN))

The top of the subject stack has no relation prefix and hence is simple; however, the top of the object stack is complex. Therefore, set string = And(form,government), top of object stack = null, and address = 0. Break string into:

relation = And
subject = form
object = government

The stacks are:

Level 4	0	And	form	government
Level 3	S	Mod	structure	----------
Level 2	S	Ofl	---------	NVN
Level 1	F	N-equivalent	---------	Ofl(structure, Ofl(Mod(party, And(rule,commu- nist)),NVN))

The top level of the subject and object stacks are both simple. The top level of the subject stack is not *'ed; therefore replace "form" by the candidate list (D) since Node(D) is the basic node which represents the concept form. Likewise, replace "government" by the candidate list (E). The relation is "And"; therefore set *pointer = *+(D)(E)+ and go to Step VII.

The stacks are:

Level 3	S	Mod	structure	*+(D)(E)+
Level 2	S	Ofl	----------	NVN
Level 1	F	N-equivalent	----------	Ofl(structure, Ofl(Mod(party, And(rule,commu- nist)),NVN))

The top level of the subject and object stacks are both simple. Since the top level of the subject stack is not *'ed, replace it with the candidate list (C,C1,C2,C3). The top of the object stack is *'ed. The top of the relation stack is not "and"; hence QAS performs Case VI identity check, which is equivalent to Case V. Set *object = D and E for each of the object candidate list in the object "and list" respectively and go to Step I, Case IV. Node (C1) has the required "Mod" relational pointer to both *objects, a subnode, Node(C), with no relational pointers, and no more than the required relational pointers. Therefore, QAS sets *pointer = *C1 and returns to the algorithm. Store *C1 in the subject stack at level 2. The stacks are:

Level 2	S	Ofl	*C1 NVN
			Ofl(structure,
Level 1	F	N-equivalent	--Ofl(structure,Ofl(Mod(party,And (rule,communist)),NVN))

The top of the subject and object stacks are both simple. The top of the subject stack is *'ed, but the top of the object stack is not. Hence, QAS replaces "NVN" with its candidate list (F). The relation is not "And". QAS, then performs a Case II identity check. Set *object = candidate object node F because it is the basic candidate node. Go to Step I, Case I. Node(C1) does not have the required "Of1" relation, but Node(C1) does have a super-node, Node(C2), which has the required relation to Node(F). Node(C2) has no other relations. QAS, therefore, sets *pointer = Node(C2). Store *C2 on the subject stack at level 1. The stacks are:

Level 1	F	N-equivalent	*C2	Of1(structure,Of1(Mod (party,And(rule,commu- nist)),NVN))

The top of the object stack is not simple, therefore, set string = Of1(structure,Of1(Mod(party,And(rule,communist)),NVN)), top of object stack = null, and address = 0). Break string into:

relation = Of1

subject = structure

object = Of1(Mod(party,And(rule,communist)),NVN)

The stacks are:

Level 2	0	Of1	structure	Of1(Mod(party,And (rule,communist)), NVN)
Level 1	F	N-equivalent	*C2	-----------------

The top of the object stack is not simple; therefore, set string = Of1(Mod(party,And(rule,communist)),NVN), top of object stacks = null, and address = 0. Break string into:

relation = Of1

subject = Mod(party,And(rule,communist))

object = NVN

The stacks are:

Level 3	0	Ofl	Mod(party,And(rule, communist))	NVN
Level 2	0	Ofl	structure	---
Level 1	F	N-equivalent	*C2	---

The top of the subject stack is not simple; therefore, set string = Mod(party,And(rule,communist)), top of subject stack = null, and address = S. Break string into:

relation = Mod

subject = party

object = And(rule,communist)

The stacks are:

Level 4	S	Mod	party	And(rule,com-munist)
Level 3	0	Ofl	-----	NVN
Level 2	0	Ofl	structure	-------------
Level 1	F	N-equivalent	*C2	-------------

The top of the object stack is not simple; therefore, set string = And(rule,communist), top of object stack = null, address = 0. Break string into:

relation = And

subject = rule

object = communist

The stacks are:

Level 5	0	And	rule	communist
Level 4	S	Mod	party	---------

Level 3	0	Ofl	----------	NVN
Level 2	0	Ofl	structure	----------
Level 1	F	N-equivalent	*C2	------- --

The top of the subject and object stacks are both simple. However, neither are *'ed or entered in the QAS dictionary. Therefore, QAS creates basic nodes, Node(H) and Node(I), and replaces "rule" and "communist" with (H) and (I) respectively. The relation is "And"; therefore, QAS stores *+(H)(I)+ on level 4 of the object stack. The stacks are:

Level 4	S	Mod	party	*+(H)(I)+
Level 3	0	Ofl	----------	NVN
Level 2	0	Ofl	structure	----------
Level 1	F	N-equivalent	*C2	----------

The top of the subject and object stacks are both simple. "Party" is not entered in the QAS dictionary; QAS therefore creates basic node, Node(G), and replaces "party" with the candidate list (G). The top of the relation stack is not "And". QAS, then, performs the Case VI identity check. QAS sets *object = H and I for each of the candidate list in the object "and list" respectively and then goes to Step I, Case IV. Node(G) does not have the required "Mod" relation; therefore, QAS creates a new node, Node(G1), with the required "Mod" relation to Node(H) and Node(I). QAS adds a subset pointer from Node(G1) to Node(G) and a superset pointer from Node(G) to Node(G1). Finally QAS sets *pointer = *G1. The union of Figures 11 and 12 represents the current memory network. Store *G1 on level

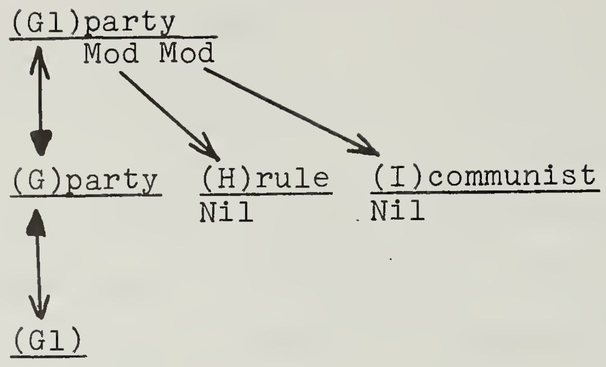


Figure 12. Memory representation of "ruling Communist party"

three of subject stack. The stacks are:

Level 3	0	Ofl	*G1	NVN
Level 2	0	Ofl	structure	---
Level 1	F	N-equivalent	*C2	---

The top of the subject and object stacks are both simple, but the top of the object stack is not *'ed. Hence, QAS replaces "NVN" with (F). Since the top of the relation stack is not "And", QAS performs the Case II identity check, QAS sets *object = F because it is the basic candidate node and goes to Step I, Case I. Node(G1) does not have the required "Ofl" relation and has no superset pointer. Hence, QAS creates a new node, Node(G2), with the "Ofl" relational pointer to Node(F), adds a subset pointer from Node(G2) to Node(G1), adds a superset pointer from Node(G1) to Node(G2), and sets *pointer = *G2. The union of Figures 11 and 13 represents the current memory network. Store *G2 on level 2 of the object stack. The stacks are:

Level 2	0	Ofl	structure	*G2

Level 1 F N-equivalent *C2 ---

The top of the subject and object stacks are both simple, but the top of the subject stack is not *'ed. Therefore, QAS replaces "structure" with (C,C1,C2,C3).

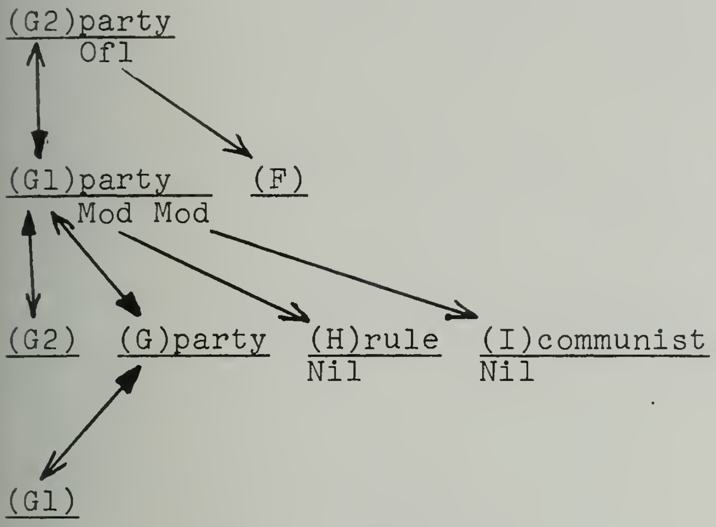


Figure 13. Memory representation of "ruling Communist party of NVN"

Since the top of the relation stack is not "And", QAS performs the Case IV identity check. QAS selects Node(C), Node(C1), and Node(C3) as candidate subjects and rejects them because they do not have the required "Of1" relational pointer. QAS selects Node(C2) which has the required relational pointer and rejects it because the object of the "Of1" relation is not Node(G2). Hence, QAS creates a new node, Node(C4), with the required "Of1" relational pointer to Node(G2), adds a subset pointer from Node(C4) to Node(C), and adds a superset pointer from Node(C) to Node(C4). Finally, QAS sets *pointer equal to *C4. The union of Figures 11 and 14 represent the current memory network.

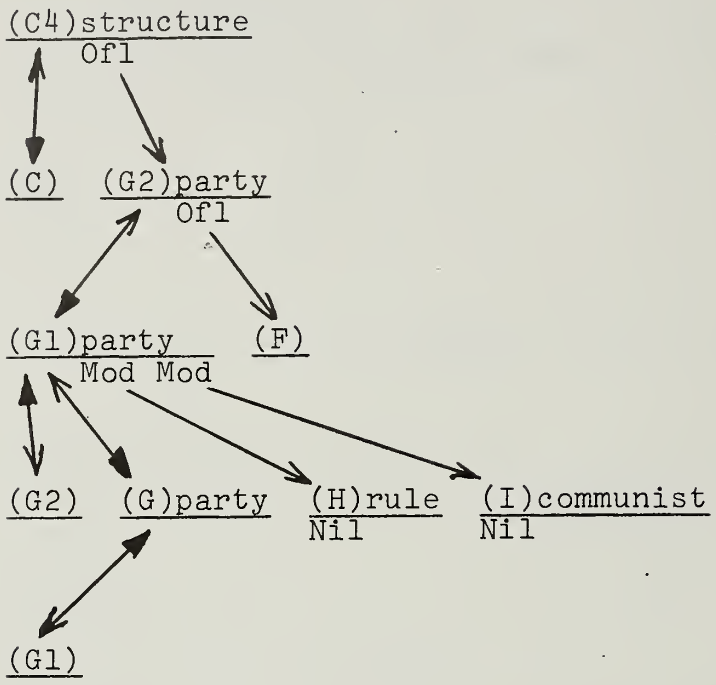


Figure 14. Memory representation of "the structure of the ruling Communist party of NVN"

Store *C4 on level one of the object stack. The stacks are:

Level 1   F   N-equivalent   *C2   *C4

Node(C2) does not have the required "N-equivalent" relation. Therefore, QAS procedes as follows to add the symmetric "N-equivalent" relation: 1) creates Node(C5) and Node(C6) with the "N-equivalent" relational pointer to each other, 2) adds subset pointers from Node(C5) to Node(C2) and from Node(C6) to Node(C4), 3) adds superset pointers from Node(C2) to Node(C5) and from Node(C4) to Node(C6), and 4) finally terminates the algorithm. Figure 15 depicts the final memory representation.

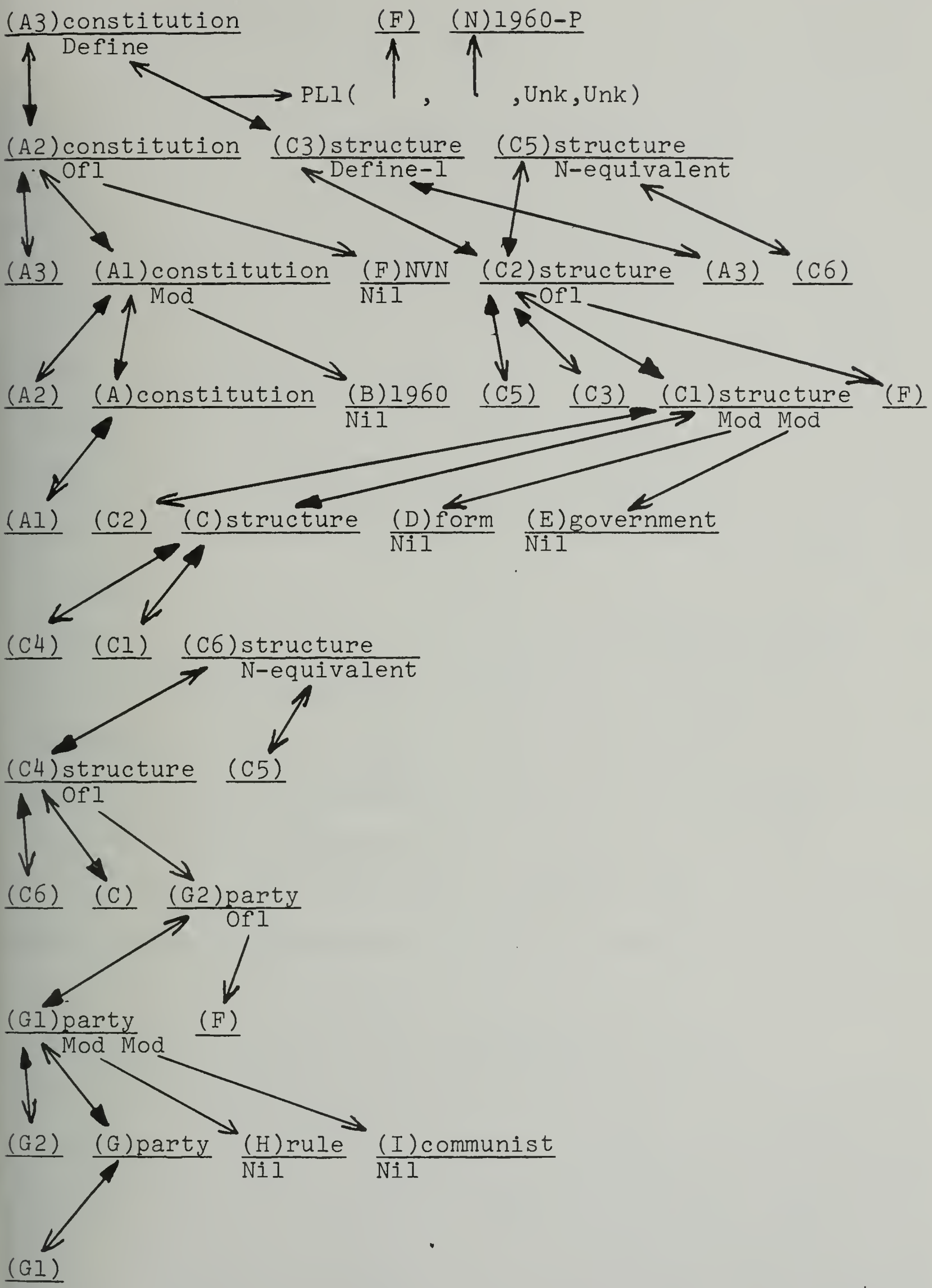


Figure 15. Memory representation of "The 1960 Constitution of NVN defines the formal governmental structure of NVN which is distinct from the structure of the ruling Communist party of NVN."

4. QUESTION ANSWERING

The encoder encodes questions for QAS in the same manner as he encodes intelligence data except that he prefixes a question specification list. QAS has two question ansering modes. The first mode answers questions which have answers explicitly represented in the memory network. The second mode answers questions which require deductions. The question specification list and both question answering modes are discussed in the paragraphs below.

4.1 Question Encoding

The encoder encodes questions utilizing the procedure described for intelligence data in Section 2.4. In addition, the encoder prefixes a question specification list, (?-Pl#,#1,#2)

The "?" causes QAS to set a flag and to modify its behavior at the times specified in the following paragraphs. PL#, if present, indicates that the desired answer is found on the main relation's property list at position #.

#1 specifies the response level desired. Level one is either 1) yes or no, or 2) (all modifiers on the subset chain of the response's subject, plus any submodifiers or Of1 relations and their objects on these modifiers) response subject (all OF1 relations on the subset chain of the response's subject, plus any submodifiers or Of1 relations and their objects on the objects and subset chains of these Ofl relations) response relation (all modifiers on the subset chain of the response's object, plus any submodifiers or Ofl relations and their objects on these modifiers) response object (all Ofl relations on the subset chain of the response's object, plus any submodifiers or Ofl relations and their objects on the objects and subset chains of these Ofl relations). N-level responses, for N greater than one, include any other relations and their objects on the subset chain of the (n-l)-level response, plus the modifiers and Ofl objects on the subset chains of these objects. #00 causes QAS to respond at the highest level possible. For example, if Node(C3) and Node(E6) of Appendix D are the response's subject and object respectively, then the level one response is "1960 constitution define form government structure of NVN".

QAS may find more than one node which represents the answer to a given question. #2 indicates the number of responses desired. #00 causes QAS to respond with all possible answers. Appendices F and G contain examples of questions and their encoded form.

4.2 Questions With Explicitly Represented Answers

QAS sets flags to indicate 1) question processing, 2) property list position of answer if applicable, 3) desired response level, and 4) desired number of responses, QAS then follows the algorithm (Section 3) for adding intelligence data to the memory network except for the following modifications.

When QAS encounters a question word or "N/A" simple subject or object, it treats it as a *subject or *object respectively and does not create a candidate list. When level one of the subject and object stacks contain *subject and *object respectively, QAS performs as follows:

```
I.  If top of subject and object stacks = *subject and *object respectively
        A.  If *subject has required relation to *object
            1.  Print "Yes"
        B.  If *subject has superset pointers
            1.  If supernode has required relation to *object
                a.  Print "Yes"
            2.  If *object has superset pointers
                a.  If supernode of *subject has required relation to supernode of *object
                    1)  Print "Yes"
                b.  Print "No"
            3.  Print "No"
        C.  Print "No"
II. If top of subject and object stack = *subject and *question word or *N/A respectively
        A.  If *subject does not have required relation
            1.  If *subject has superset pointers
                a.  If supernode has the required relation
                    1)  Subject of the desired response = supernode
```

2) If PL# tag is present

a) Go to Step IV

3) Print desired response

b. Deductive procedures are employed at this point. See Section 4.3.

2. Go to Step II.A.1.b.

B. Subject of desired response = *subject

C. If PL# tag is present

1. Go to Step IV

D. Print desired response

II. If top of subject and object = *question word or *N/A and *object respectively

A. If *object does not have required converse or symmetric relation

1. If *object has superset pointers

a. If supernode has the required converse or symmetric relation

1) Subject of the desired response = the node which is the object of the converse or symmetric relation on supernode

2) If PL# tag is present

a) Go to Step IV

3) Print desired response

b. Go to Step II.A.1.b.

B. Subject of desired response = the node which is the object of the converse or symmetric relation on *object

```
        C.  If PL# tag is present
            1.  Go to Step IV
        D.  Print desired response
 IV.  If PL1
        A.  Print desired response, plus "in", plus desired response for Node(position one pointer)
  V.  If PL2
        A.  Print desired response, plus "during", plus desired response for Node(position two pointer)
 VI.  If PL3
        A.  Print desired response, plus "by means of", plus desired response for Node(position three  pointer)
VII.  If PL4
        A.  Print desired response, plus "for the purpose of", plus desired response for Node(position four pointer)
```

Appendix F contains examples of questions with explicitly represented answers, the encoded questions, and QAS responses to the questions.

4.3 Questions Which Require Deduction

When QAS cannot find a node with the required pointer while operating in the question answering mode, QAS must employ deductive heuristics. For example, QAS must employ deductive heuristics at each point where it would normally create a new node in the algorithm of Section 3 and at the points specified in the procedure of Section 4.2.

QAS prints a message which informs the user of the heuristic(s) used to obtain an answer. Deductive heuristics may lead to zero, one, or many anssers. QAS must keep track of all such possible answers. At stack levels greater than one, the possible answers are answers to subquestions or subgoals of the original question which may aid in determining the desired response. As QAS pops the stacks, some answers to higher level subgoals may be of no use in determining the answers to the present level subgoal and are thusly discarded. The deductive heuristics include:

If *subject does not have the required relation, but does have a subnode with the required relation to *object or *object is on the subset chain of the required relation accept relation(subnode,relation's object) as the desired concept.

If *subject has the required relation, and *object is on the subset chain of the required relation, accept relation (*subject, relation's object) as the desired concept.

If *subject has required relation to one or more elements of a object candidate list, accept relation(*subject, relation's object) as the desired concept.

If candidate subject has required relational pointer to *object or *object is on the subset chain of the relation's object, accept relation(candidate subject,relation's object) as the desired concept whether or not candidate subject has a subnode with relational pointers.

If candidate subject has the required relational pointer to *object or *object is on the subset chain of the relation's object, accept relation(candidate subject, relation's object) as the desired concept whether or not candidate subject has more relational pointers than the required relation.

If candidate subject or *subject has the required relational pointer and *object is the object of Ofl for some node on the subset chain of the required relation's object, accept relation(*subject or candidate subject,relation's object) as the desired concept.

If the required relational Mod or Ofl pointer is not present, check to see if Ofl or Mod is present respectively.

If required relational pointer is present and transitive and *object is not the object of the required relation, search for *object as the object node or on the subset chains of any node which can be reached by following the transitive relational pointer or subset pointers. If *object is found, accept relation(subject,relation's object) as the desired concept.

If required relational pointer is not present, search for equivalent relational pointers.

If *subject does not have the required relation, but does have an immediate supernode with the required relation to *object or *object is on the subset chain of the required relation's object, accept relation(supernode,relation's object) as the desired concept even if supernode has additional unrequired relational pointers.

Appendix G contains examples of questions which require deduction to answer, the encoded questions, and QAS responses to the questions.

LIST OF REFERENCES

1. Alexander, B., "A question-Answering Program For Simple Kernel Sentences (QUE 2)," Technical Report No. NL-5, March 1971, DCS and Computer-Assisted Instruction Laboratory, Univ. of Texas, Austin, Texas, pp. 1-30.

2. Biss, K. O., Chien, R. T., and Stahl, F. A., "A Data Structure for Cognitive Information Retrieval," Report R-493, October, 1970, CSL, Univ. of Illinois, Urbana, Illinois.

3. Biss, K. O., Chien, R. T., and Stahl, F. A., "R2 - A Natural Language Question-Answering System," Report R-500, January, 1971, CSL, Univ. of Illinois, Urbana, Illinois.

4. Black, F., "A Deductive Question-Answering System," Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, Massachusetts, 1968, pp. 354-402.

5. Bobrow, D. G., "Natural Language Input For A Computer Problem-Solving System," Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, Massachusetts, 1968, pp. 135-215.

6. Bobrow, D. G. "Problems In Natural Language Communication With Computers," Project No. 8668, Scientific Report No. 5, August, 1966, Bolt Beranck and Newman, Inc., 50 Moulton Street, Cambridge, Massachusetts.

7. Bobrow, D. G., Fraser, J. B., and Quillian, R. M., "Automated Language Processing," Annual Review of Information Science and Technology, C. A. Cuadis (Ed.), Vol. 2, Interscience Publishers, New York, 1967, pp. 161-186.

8. Chang, Y., "The Implementation Of A Relational Document Retrieval System," Report R-489, September, 1970, CSL, Univ. of Illinois, Urbana, Illinois.

9. Chien, R. T., Ray, S. R., and Stahl, F. A., "ISL - A New Programming Language For Information Retrieval," Report R-449, December, 1969, CSL, Univ. of Illinois, Urbana, Illinois.

10. Craig, J. A., Berezner, S. C., Carney, H. C., and Longyear, C. R., "Deacon: Direct English Access And Control," Proceedings - Fall Joint Computer Conference, 1966, Vol. 29, Spartan Books, New York, 1966, pp. 365-380.

11. Esser, N. M. Jr., "A Relational Structure For Document Retrieval In Coding Theory," Report R-469, May, 1970, CSL, Univ. of Illinois, Urbana, Illinois.

12. Fairthorne, R. A., Towards Information Retrieval, Butterworths, London, 1961.

13. Floyd, R. W., "On Syntactic Analysis And Operator Precedence," Scientific Report No. 1, September, 1962, Computer Associates, Inc., 44 Winn Street, Woburn, Massachusetts, pp. 1-31.

14. Green, B. G. Jr., Wolf, A. K., Chomsky, C., and Laughery, K., "Baseball: An Automatic Question Answerer," Computers and Thought, E. A. Feigenbaum and J. Feldman (Ed.), McGraw-Hill Book Co., New York, 1963, pp. 207-216.

15. Hays, D. G., "Automatic Language - Data Processing," Computer Applications In The Behavioral Sciences, H. Borko (Ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962, pp. 395-423.

16. Jansen, J. M. Jr., "Phrase Dictionary Construction Mehtods For The R2 Information Retrieval System," Report R-447, December, 1969, CSL, Univ. of Illinois, Urbana, Illinois.

17. Katz, J. J., "Analyticity And Contradiction In Natural Language," The Structure of Language, J. A. Fodor and J. J. Katz (Ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964, pp. 519-543.

18. Katz, J. J., Fodor, J. A., "The Structure Of A Semantic Theory," The Structure of Language, J. A. Fodor and J. J. Katz (Ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 479-518.

19. Kelley, K. C., Ray, S. R., and Stahl, F. A., "ISL - A String Manipulating Language," Report R-407, February, 1969, CSL, Univ. of Illinois, Urbana, Illinois.

20. Kent, Allen, Textbook On Mechanized Information Retrieval, Interscience Publishers, New York, 1966.

21. Lancaster, F. W., Information Retrieval Systems, John Wiley and Sons, Inc., New York, 1968.

22. Lindsay, R. K., "Inferential Memory As The Basis Of Machines Which Understand Natural Language," Computers and Thought, E. A. Feigenbaum and J. Feldman (Eds.), McGraw-Hill Book Co., Inc., New York, 1963, pp. 217-233.

23. McCarthy, J., "Programs With Common Sense," Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, Massachusetts, 1968, pp. 403-418.

24. Mills, R. G., "Man - Machine Communication And Problem Solving," Annual Review of Information Science and Technology, Vol. 2, Interscience Publishers, New York, 1967, pp. 223-255.

25. Quillian, R. M., "Semantic Memory, Report No. 2, Bolt Beranek and Newman, Cambridge, Massachusetts, 1966, pp. 1-222.

26. Quillian, R. M., "The Teachable Language Comprehender: A Simulation Program And Theory Of Language," Report No. 10, Bolt Beranek and Newman, Cambridge, Massachusetts, 1969, pp. 1-60.

27. Raphael, B., "Sir: Semantic Information Retrieval," Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, Massachusetts, 1968, pp. 33-134.

28. Richards, I. A., Basic English and Its Uses, W. W. Norton and Co., Inc., 1943, New York.

29. Rosenbaum, P. S., and Blair, F., "Sepcification And Utilization Of A Transformational Grammar," Project 4641, Task 464102, October, 1966, International Business Machines Corporation, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York.

30. Schultz, J. A., and Bielby, W. T., "An Algorithm For The Syntactic Analysis In The R2 Information System," Report R-494, October, 1970, CSL, Univ. of Illinois, Urbana, Illinois.

31. Simmons, R. F., "Answering English Question By Computer: A Survey," Communications of the ACM, A. G. Oettinger (Ed.), Vol. 8, No. 1, January, 1965, pp. 53-70.

32. Simmons, R. F., "Automated Language Processing," Annual Review of Information Science and Technology, C. A. Cuadia (Ed.), Vol. 1, Interscience Publishers, 1966, pp. 137-169.

33. Simmons, R. F., "Natural Language Question - Answering Systems: 1969," Communications of The ACM, D. G. Bobrow, (Ed.), Vol. 13, No. 1, January, 1970, pp. 15-30.

34. Simmons, R. F., "Synthex," Conversational Computers, W. D. Orr (Ed.), John Wiley and Sons, Inc., New York, 1968, pp. 121-127.

35. Simmons, R. F., "Syntex: Toward Computer Synthesis of Human Language Behavior," Computer Applications In The Behavioral Sciences, H. Borko (Ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962, pp. 361-393.

36. Simmons, R. F., and Bruce, B. C., "Some Relations Between Predicate Calculus And Semantic Net Representations of Discourse," Technical Report No. NL-2, DCS, Univ. of Texas, Austin, Texas, June, 1971.

37. Simmons, R. F., Burger, J. E., and Long, R. E., "An Approach Toward Answering English Questions From Text," Proceedings - 1966 Fall Joint Computer Conference, Vol. 29, Spartan Books, New York, 1966, pp. 357-363.

38. Simmons, R. F., Burger, J. F., and Schwarcz, R. M., "A Computational Mode of Verbal Understanding," Proceedings of the AFIPS 1968 Fall Joint Computer Conference, Vol. 33, Part I, Thompson Book Co., Washington, D. C., 1968, pp. 441-456.

39. Simmons, R. F., Schwarcz, R. M., and Burger, J. F., "A Deductive Question - Answerer For Natural Language Inference," Communication of the ACM, Vol. 13, No. 3, D. G. Bobrow (Ed.), March, 1970, pp. 167-183.

40. Shapiro, S. C., and Woodmansee, G. H., "A Net Structure Based Relational Question Answer: Description and Examples," DCS, Univ. of Wisconsin, Madison, Wisconsin, pp. 325-346.

41. Smith, H. H., Bernier, D. W., et al., Area Handbook for North Vietnam, DA Pamphlet No. 550-57, U.S. Government Printing Office, June, 1967, Washington, D.C.

42. Thompson, F. B., "Deacon - Type Query Systems," Conversational Computers, W. D. Orr (Ed.), John Wiley and Sons, Inc., New York, 1968, pp. 128-134.

43. Thompson, F. B., "English For The Computer," Proceedings - 1966 Fall Joint Computer Conference, Vol. 29, Spartan Books, 1966, pp. 349-356.

APPENDIX A

QAS RELATIONS

Relations:

Unary

(1)	Country(x) x is a country	name of country
(2)	Name(x) x is a name	name of person
(3)	Party(x) x is a party	name of political party
(4)	Place(x) x is a place	geographic location - place name
(5)	Time (x) x is a time	written as year, month, day, time (24 hour clock, GMT) ex: 197107291945 or 19710729xxxx "x" implies not specified)
(6)	Purpose(x) x is a purpose	an object or result aimed at
(7)	Agent(x) x is an agent	something that produces or is capable of producing an effect, i.e., the purpose in (6)

Binary

(8)	Mod(x,y) x is modified by y	y is an adjective which modifies x
(9)	Define(x,y) x defines y	to make distinct in outline - active verb
(10)	Ofl(x,y) x belongs to y	used as function word to indicate belonging or possessive relationship - transitive

(11)	Distinct(x,y) x is distinct from y	distinguished from others - symmetric
(12)	Equivalent(x,y) x is equivalent to y	corresponding or virtually identical especially in effect or function - transitive - symmetric
(13)	Extends-to(x,y) x extends to y	to span an interval of distance, space, or time - active verb
(14)	Component(x,y) x is a component of y	a constituent part-transitive
(15)	N-component(x,y) x is not a component of y	not a constituent part
(16)	Supervise(x,y) x supervises y	oversee - active verb - transitive
(17)	Control(x,y) x controls y	to exercise restraining or directing influence over - to have power over - active verb - transitive
(18)	Some(x,y) some x are y	being one, a part, or an unspecified number of something (as a class or group) named or implied - transitive, symmetric (when x and y are nouns)
(19)	Of2(x,y) x of y	used as a function word to indicate a characteristic or distinctive quality or possession
(20)	Inl(x,y) x in y	used as a function word to indicate inclusion
(21)	Hold(x,y) x holds y	to have earned or been appointed, promoted, or elected to and now occupy (as an office) - active verb

(22)	Counterpart(x,y) x is a counterpart of y	one remarkably similar to another - one having the same function or characteristics as another - symmetric
(23)	And (x_1,x_2) x_1 and x_2	used as a function word to show a conjunctive relationship between x_1 and x_2
(24)	Or (x_1,x_2) x_1 and x_2	used as a function word to show a disjunctive relationship between x_1 and x_2

Relational Relations

(25) $\mathrm{Ofl}(x,y)\ \&\ \mathrm{Ofl}(y,z) \rightarrow \mathrm{Ofl}(x,z)$

(26) $\mathrm{Equivalent}(x,y) = \text{N-distinct}(x,y)$

(27) $\mathrm{Mod}(x,y) = \mathrm{Ofl}(x,y)$

APPENDIX B

ORIGINAL TEXT BEFORE ENCODING INTO QAS FORMAT

The formal governmental structure, as defined in the 1960 Constitution, is distinct from that of the ruling Communist party, known as the Lao Dong Party, which is not mentioned specifically in the Constitution as an integral component of the governmental system. Nevertheless, the formal machinery of government is supervised and controlled by the Party, particularly when Party leader and government official are the same person.

The government is controlled by the Party through an interlocking system of parallel hierarchies which extend down into the lowest territorial units. At the highest level, ranking members of the Party's Central Committee and Political Bureau (Politburo) hold key positions in the central government. Various functional departments or bureaus of the Central Committee in turn control and supervise the operation of their counterparts in the government, usually called ministries and commissions. At the local level the Party's lower committees similarly control the activities of local government machinery.

APPENDIX C

SUBSENTENCES AND THEIR ENCODED FORM

Paragraph 1, Sentence 1

Subsentence 1

The 1960 Constitution of NVN defines the formal governmental structure of NVN.

Encoded as:

Define/NVN,1960-P,Unk,Unk/(Of1(Mod(constitution,1960), NVN),Of1(Mod(structure,And(form,government)),NVN))

Subsentence 2

The formal governmental structure of NVN is distinct from the structure of the ruling Communist party of NVN.

Encoded as:

N-equivalent(Of1(Mod(structure,And(form,government)), NVN),Of1(structure,Of1(Mod(party,And(rule,communist)),NVN)))

Subsentence 3

The ruling Communist Party of NVN is equivalent to the Lao Dong Party of NVN.

Encoded as:

Equivalent(Of1(Mod(party,And(rule,communist)),NVN),Of1 (Mod(party,Lao Dong),NVN))

Subsentence 4

The Lao Dong Party of NVN is not an integral component of the governmental system of NVN.

Encoded as:

N-component(Of1(Mod(party,Lao Dong),NVN),Of1(Mod(system, government)NVN))

Paragraph 1, Sentence 2

Subsentence 5

The Lao Dong Party of NVN controls the formal machinery of the government of NVN.

Encoded as:

Control/NVN,1960-P,Unk,Unk/(Of1(Mod(party,Lao Dong),NVN), Of1(Mod(machine,form),Of1(government,NVN)))

Subsentence 6

The Lao Dong Party of NVN supervises the formal machinery of the government of NVN.[1]

Encoded as:

Supervise/NVN,1960-P,Unk,Unk/(Of1(Mod(party,Lao Dong), NVN),Of1(Mod(machine,form),Of1(government,NVN)))

Subsentence 7

Some leaders of the Lao Dong Party of NVN are government officials of NVN.

Encoded as:

Some(Of1(leaders,Of1(Mod(party,Lao Dong),NVN)),Of1(Mod (official,government),NVN))

Paragraph 2, Sentence 1

Subsentence 8

The Lao Dong Party of NVN controls the government of NVN through an interlocking system of parallel hierarchies.

Encoded as:

Control/NVN,1960-P,Of2(Mod(system,interlock),Mod(hierarchy parallel)),Unk/(Of1(Mod(party,Lao Dong),NVN),Of1(Government,NVN))

---

[1]As yet, QAS does not provide for Connective relations on relations.

Subsentence 9

The interlocking system of parallel hierarchies extends down into the lowest territorial units.

Encoded as:

```
        Extend-to/NVN,1960-P,Unk,Unk/(Of2(Mod(system,interlock),
Mod(hierarchy,parallel)),Mod(unit,and(low,territory)))
```

Paragraph 2, Sentence 2

Subsentence 10

Ranking members of the Central Committee and Politburo of the Lao Dong Party of NVN hold key positions in the central government of NVN.

Encoded as:

```
        Hold/NVN,1960-P,Unk,Unk/(Of1(Mod(member,rank),Of1(And
(Mod(committee,central),politburo),Of1(Mod(party,Lao Dong),
NVN))),In1(Mod(position,key),Of1(Mod(government,central),NVN)))
```

Paragraph 2, Sentence 3

Subsentence 11

The functional departments of the Central Committee of the Lao Dong Party of NVN are equivalent to the bureaus of the Cnetral Committee of the Lao Dong Party of NVN.

Encoded as:

```
        Equivalent(Of1(Mod(department,function),Of1(Mod(committee,
central),Of1(Mod(party,Lao Dong),NVN))),Of1(bureau,Of1(Mod
(committee,central)Of1(Mod(party,Lao Dong),NVN))))
```

Subsentence 12

The bureaus of the Central Committee of the Lao Dong Party of NVN Control the operation of their counterparts in the government of NVN.

Encoded as:

```
        Control/NVN,1960-P,Unk,Unk/(Of1(bureau,Of1(Mod(committee,
central),Of1(Mod(party,Lao Dong),NVN))),Of1(operation,In1(Of1
(counterpart,Of1(bureau,Of1(Mod(committee,central),Of1(Mod(party,
Lao Dong),NVN)))),Of1(government,NVN))))
```

Subsentence 13

The bureaus of the Central Committee of the Lao Dong Party of NVN supervise the operation of their counterparts in the government of NVN.

Encoded as:

Supervise/NVN,1960-P,Unk,Unk/(Ofl(bureau,Ofl(Mod(committee central),Ofl(Mod(party,Lao Dong),NVN))),Ofl(operation,Inl(Ofl (counterpart,Ofl(bureau,Ofl(Mod(committee,central),Ofl(Mod(party, Lao Dong),NVN)))),Ofl(government,NVN))))

Subsentence 14

The counterparts of the bureaus of the Central Committee of the Lao Dong Party of NVN in the government of NVN are called ministries and commissions.

Encoded as:

Equivalent(Ofl(counterpart,Ofl(bureau,Ofl(Mod(committee, central),Ofl(Mod(party,Lao Dong),NVN)))),Inl(And(ministry,com-mission),Ofl(government,NVN)))

Paragraph 2, Sentence 4

Subsentence 15

The lower committees of the Lao Dong Party of NVN control the activities of local government machinery.

Encoded as:

Control/NVN,1960-P,Unk,Unk/(Ofl(Mod(committee,low),Ofl (Mod(party,Lao Dong),NVN)),Ofl(activity,Mod(machine,And(local government))))

APPENDIX D

MEMORY REPRESENTATION OF THE TEXT

Relation Tag	Relation
R1	Mod
R2	Define
R3	Of1
R4	And
R5	Component
R6	Control
R7	Supervise
R8	Some
R9	Of2
R10	Extend-to
R11	Hold
R12	In1
R13	Equivalent

Dictionary

1. NVN(A)
2. 1960-P(B)
3. constitution(C,C1,C2,C3)
4. 1960(D)
5. structure(E,E1,E2,E3,E4,E5)
6. form(F)
7. government(G,G1,G2,G3,G4)
8. rule(H)
9. communist(I)
10. party(J,J1,J2,J3,J4,J5,J6,J7,J8,J9,J10)
11. Lao Dong(K)
12. system(L,L1,L2,L3,L4,L5)
13. machine(M,M1,M2,M3,M4,M5)
14. leader(N,N1,N2)
15. official(O,O1,O2,O3)
16. hierarchy(P,P1)
17. parallel(Q)
18. interlock(R)
19. low(S)
20. territory(T)
21. unit(U,U1,U2)
22. member(V,V1,V2,V3)
23. rank(W)
24. committee(X,X1,X2,X3,X4,X5)
25. central(Y)

26. politburo(Z,Z1)
27. position(AA,AA1,AA2,AA3)
28. key(BB)
29. department(CC,CC1,CC2,CC3)
30. function(DD)
31. bureau(EE,EE1,EE2,EE3,EE4)
32. counterpart(FF,FF1,FF2,FF3)
33. operation(GG,GG1,GG2,GG3)
34. ministry(HH,HH1,HH2)
35. commission(II,II1,II2)
36. activity(JJ,JJ1,JJ2)
37. local(KK)

Memory Network

```
A(Dic(1),Sub(Nil),Sup(Nil))
B(Dic(2),Sub(Nil),Sup(Nil))
C(Dic(3),Sup(Nil),Sup(C1))
D(Dic(4),Sub(Nil),Sup(Nil))
E(Dic(5),Sub(Nil),Sup(E1,E3))
F(Dic(6),Sub(Nil),Sup(Nil))
G(Dic(7),Sub(Nil),Sup(G1,G3))
C1(Dic(3),Sub(C),Sup(C2),R1(D))
C2(Dic(3),Sub(C1),Sup(C3),R3(A))
E1(Dic(5),Sub(E),Sup(E2),R1(F,G))
E2(Dic(5),Sub(E1),Sup(E5,E6),R3(A))
C3(Dic(3),Sub(C2),Sup(Nil),R2(E6-(A,B,Unk,Unk)))
H(Dic(8),Sub(Nil),Sup(Nil))
I(Dic(9),Sub(Nil),Sup(Nil))
J(Dic(10),Sub(Nil),Sup(J1,J3)
J1(Dic(10),Sub(J),Sup(J2),R1(H,I))
J2(Dic(10),Sub(J1),Sup(J5),R3(A))
E3(Dic(5),Sub(E),Sup(E4),R3(J2))
E4(Dic(5),Sub(E3),Sup(Nil),N-R13(E5))
E5(Dic(5),Sub(E2),Sup(Nil),N-R13(E4))
K(Dic(11),Sub(Nil),Sup(Nil))
J3(Dic(10),Sub(J),Sup(J4),R1(K))
J4(Dic(10),Sub(J3),Sup(J6,J7,J8,J9,J10),R3(A))
J5(Dic(10),Sub(J2),Sup(Nil),R13(J6))
J6(Dic(10),Sub(J4),Sup(Nil),R13(J5))
L(Dic(12),Sub(Nil),Sup(L1,L3))
L1(Dic(12),Sub(L),Sup(L2),R1(G))
L2(Dic(12),Sub(L1),Sup(Nil),R3(A))
J7(Dic(10),Sub(J4),Sup(Nil),N-R5(L2))
E6(Dic(5),Sub(E2),Sup(Nil),R2-1(C3-(A,B,Unk,Unk)))
M(Dic(13),Sub(Nil),Sup(M1,M5))
M1(Dic(13),Sub(M),Sup(M2),R1(F))
G1(Dic(7),Sub(G),Sup(G2),R3(A))
M2(Dic(13),Sub(M1),Sup(M3,M4),R3(G1))
```

```
J8(Dic(10),Sub(J4),Sup(Nil),R6(M3-(A,B,Unk,Unk)))
M3(Dic(13),Sub(M2),Sup(Nil),R6-1(J8-A,B,Unk,Unk)))
J9(Dic(10,Sub(J4),Sup(Nil),R7(M4-(A,B,Unk,Unk)))
M4(Dic(13),Sub(M2),Sup(Nil),R7-1(J9-(A,B,Unk,Unk)))
N(Dic(14),Sub(Nil),Sup(N1))
O(Dic(15),Sub(Nil),Sup(O1)
N1(Dic(14),Sub(N),Sup(N2),R3(J4))
O1(Dic(15),Sub(O),Sup(O2),R1(G))
O2(Dic(15),Sub(O1),Sup(O3),R3(A))
N2(Dic(14),Sub(N1),Sup(Nil),R8(O3))
O3(Dic(15),Sub(O2),Sup(Nil),R8(N2))
P(Dic(16),Sub(Nil),Sup(P1))
Q(Dic(17),Sub(Nil),Sup(Nil))
R(Dic(18),Sub(Nil),Sup(Nil))
L3(Dic(12,Sub(L),Sup(L4),R1(R))
P1(Dic(16),Sub(P),Sup(Nil),R1(Q))
L4(Dic(12),Sub(L3),Sup(L5),R9(P1))
J10(Dic(10),Sub(J4),Sup(Nil),R6(G2-(A,B,L4,Unk)))
G2(Dic(7),Sub(G1),Sup(Nil),R6-1(J10-(A,B,L4,Unk)))
S(Dic(19),Sub(Nil),Sup(Nil))
T(Dic(20),Sub(Nil),Sup(Nil))
U(Dic(21),Sub(Nil),Sup(U1))
U1(Dic(21),Sub(U),Sup(U2),R1(S,T))
L5(Dic(12),Sub(L4),Sup(Nil),R10(U2-(A,B,Unk,Unk)))
U2(Dic(21),Sub(U1),Sup(Nil),R10-1(L5-(A,B,Unk,Unk)))
V(Dic(22),Sub(Nil),Sup(V1))
W(Dic(23),Sub(Nil),Sup(Nil))
X(Dic(24),Sub(Nil),Sup(X1,X3))
Y(Dic(25),Sub(Nil),Sup(Nil))
Z(Dic(26),Sub(Nil),Sup(Z1))
AA(Dic(27),Sub(Nil),Sup(AA1))
BB(Dic(28),Sub(Nil),Sup(Nil))
V1(Dic(22),Sub(V),Sup(V2),R1(W))
X1(Dic(24),Sub(X),Sup(X2),R1(Y))
X2(Dic(24),Sub(X1),Sup(Nil),R3(J4))
Z1(Dic(26),Sub(Z),Sup(Nil),R3(J4))
V2(Dic(22),Sub(V1),Sup(V3),R3(X2,Z1))
AA1(Dic(27),Sub(AA),Sup(AA2),R1(BB))
G3(Dic(7),Sub(G),Sup(G4),R1(Y))
G4(Dic(7),Sub(G3),Sup(Nil),R3(A))
AA2(Dic(27),Sub(AA1),Sup(AA3),R12(G4))
V3(Dic(22),Sub(V2),Sup(Nil),R11(AA3-(A,B,Unk,Unk)))
AA3(Dic(27),Sub(AA2),Sup(Nil),R11-1(V3-(A,B,Unk,Unk)))
CC(Dic(29),Sub(Nil),Sup(CC1))
DD(Dic(30),Sub(Nil),Sup(Nil))
EE(Dic(31),Sub(Nil),Sup(EE1))
CC1(Dic(29),Sub(CC),Sup(CC2),R1(DD))
CC2(Dic(29),Sub(CC1),Sup(CC3),R3(X2))
EE1(Dic(31),Sub(EE),Sup(EE2,EE3,EE4),R3(X2))
CC3(Dic(29),Sub(CC2),Sup(Nil),R13(EE2))
EE2(Dic(31),Sub(EE1),Sup(Nil),R13(CC3))
FF(Dic(32),Sub(Nil),Sup(FF1))
```

```
GG(Dic(33),Sub(Nil),Sup(GG1))
FF1(Dic(32),Sub(FF),Sup(FF2,FF3),R3(EE1))
FF2(Dic(32),Sub(FF1),Sup(Nil),R12(G1))
GG1(Dic(33),Sub(GG),Sup(GG2),R3(FF2))
EE3(Dic(31),Sub(EE1),Sup(Nil),R6(GG2-(A,B,Unk,Unk)))
GG2(Dic(33),Sub(GG1),Sup(Nil),R6-1(EE3-(A,B,Unk,Unk)))
EE4(Dic(31),Sub(EE1),Sup(Nil),R7(GG4-(A,B,Unk,Unk)))
GG3(Dic(33),Sub(GG1),Sup(Nil),R7-1(EE4-(A,B,Unk,Unk)))
HH(Dic(34),Sub(Nil),Sup(HH1))
II(Dic(33),Sub(Nil),Sup(II1))
HH1(Dic(34),Sub(HH),Sup(HH2),R12(G1))
II1(Dic(35),Sub(II),Sup(II2),R12(G1))
FF3(Dic(32),Sub(FF1),Sup(Nil),R13(HH2,II2))
HH2(Dic(34),Sub(HH1),Sup(Nil),R13(FF3))
II2(Dic(35),Sub(II1),Sup(Nil),R13(FF3))
JJ(Dic(36),Sub(Nil),Sup(JJ1))
KK(Dic(37),Sub(Nil),Sup(Nil))
X3(Dic(24),Sub,(X),Sup(X4),R1(S))
X4(Dic(24),Sub(X3),Sup(X5),R3(J4))
M5(Dic(13),Sub(M),Sup(Nil),R1(KK,G))
JJ1(Dic(36),Sub(JJ),Sup(JJ2),R3(M5))
X5(Dic(24),Sub(X4),Sup(Nil),R6(JJ2-(A,B,Unk,Unk)))
JJ2(Dic(36),Sub(JJ1),Sup(Nil),R6-1(X5-(A,B,Unk,Unk)))
```

APPENDIX E

ALGORITHM RESULTS WHEN APPLIED TO DATA OF APPENDIX C

Relation Tag	Relation
R1	Mod
R2	Define
R3	Of1
R4	And
R5	Component
R6	Control
R7	Supervise
R8	Some
R9	Of2
R10	Extend-to
R11	Hold
R12	In1
R13	Equivalent

Dictionary

1. NVN(A)
2. 1960-P(B)
3. constitution(C,C1,C2,C3)
4. 1960(D)
5. structure(E,E1,E2,E3,E4,E5)
6. form(F)
7. government(G,G1,G2,G3,G4)
8. rule(H)
9. communist(I)
10. party(J,J1,J2,J3,J4,J5,J6,J7,J8,J9,J10)
11. Lao Dong(K)
12. system(L,L1,L2,L3,L4,L5)
13. machine(M,M1,M2,M3,M4,M5)
14. leader(N,N1,N2)
15. official(O,O1,O2,O3)
16. hierarchy(P,P1)
17. parallel(Q)
18. interlock(R)
19. low(S)
20. territory(T)
21. unit(U,U1,U2)
22. member(V,V1,V2,V3)
23. rank(W)
24. committee(X,X1,X2,X3,X4,X5)
25. central(Y)

26. politburo(Z,Z1)
27. position(AA,AA1,AA2,AA3)
28. key(BB)
29. department(CC,CC1,CC2,CC3)
30. function(DD)
31. bureau(EE,EE1,EE2,EE3,EE4)
32. counterpart(FF,FF1,FF2,FF3)
33. operation(GG,GG1,GG2,GG3)
34. ministry(HH,HH1,HH2)
35. commission(II,II1,II2)
36. activity(JJ,JJ1,JJ2)
37. local(KK)

Subsentence 1

```
R2/A,B,Unk,Unk/R3(R1((C),(D)),(A)),R3(R1((E),R4((F),(G))),(A)))
                  *C1                          *+(F)(G)+
               --------------------
               *C2
                                      ---------------------
                                      *E1
                                   ----------------------------
                                   *E2
----------------------------------------------------------------
*C3
```

Subsentence 2

```
N-R13(R3(R1((E-E2,E6),R4((F),(G))),(A)),R3((E-E2,E6),R3(R1((J),R4((H),(I))),(A)))
                      *+(F)(G)+                                 *  (H)(I)+
         --------------------------                     --------------------
         *E1                                            *J1
      -----------------------------                  --------------------------
      *E2                                            *J2
                                         ----------------------------------------
                                         *E3
----------------------------------------------------------------------------------
*E5
```

Subsentence 3

```
R13(R3(R1((J-J2),R4((H),(I))),(A)),R3(R1((J-J2),(K)),(A)))
                 *+(H)(I)+             *J3
       -----------------------     ----------------------
       *J1                         *J4
    ------------------------------
    *J2
------------------------------------------------------------
*J5
```

Subsentence 4

```
N-R5(R3(R1((J-J6),(K)),(A)),R3(R1((L),(G)),(A)))
        *J3                    *L1
     ----------------------  --------------------
     *J4                     *L2
-------------------------------------------------
*J7
```

Subsentence 5

```
R6/A,B,Unk,Unk/(R3(R1((J-J7),(K)),(A)),R3(R1((M),(F)),R3((G),(A))))
                   *J3                    *M1         *G1
                ______________________ ____________________________
                *J4                    *M2
___________________________________________________________________
*J8
```

Subsentence 6

```
R7/A,B,Unk,Unk/(R3(R1((J-J8),(K)),(A)),R3(R1(M-M3),(F)),R3((G-G1),(A))))
                   *J3                    *M1            *G1
                ______________________ _________________________________
                *J4                    *M2
________________________________________________________________________
*J9
```

Subsentence 7

```
R8(R3((N),R3(R1((J-J9),(K)),(A))),R3(R1((0),(G-G1)),(A)))
             *J3                     *01
          ______________________  _______________________
          *J4                     *02
   ______________________________
   *N1
_________________________________________________________
*N2
```

Subsentence 8

```
R6/A,B,R9(R1((L-L2),(R)),R1((P),(Q))),Unk/(R3(R1((J-J9),(K)),(A)),R3((G-G1),(A)))
          *L3            *P1                  *J3                  *G1
       ______________________________     ______________________
       *L4                                *J4
_________________________________________________________________________________
*J10
```

Subsentence 9

```
R10/A,B,Unk,Unk/(R9(R1((L-L4),(R)),R1((P-P1),(Q))),R1((U),R4((S),(T))))
                    *L3            *P1                    *+(S)(T)+
                 _________________________________  ___________________
                 *L4                                *U1
_______________________________________________________________________
*L5
```

Subsentence 10

```
R11/A,B,Unk,Unk/(R3(R1((V),(W)),R3(R4(R1((X),(Y)),(Z)),R3(R1((J-J10),(K)),(A)))),R12(R1((AA),(BB)),R3(R1((G-G2),(Y)),(A))))
                    *V1                 *X1                *J3                          *AA1               *G3
                                     *+*X1(Z)+           *J4                                              *G4
                                   *+*X2*Z1+                                         *AA2
                  *V2
*V3
```

Subsentence 11

```
R13(R3(R1((CC),(DD)),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A)))),R3((EE),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A)))))
        *CC1             *X1                 *J3                                 *X1                *J3
                                          *J4                                                    *J4
                       *X2                                                *X2
     *CC2                                                         *EE1
*CC2
```

Subsentence 12

```
R6/A,B,Unk,Unk/R3((EE-EE2),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A)))),R3((GG),R12(R3((FF),R3((EE-EE2),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A))))),R3((G-G4),(A)))))
                               *X1                *J3                                                          *X1                *J3                       *G1
                                               *J4                                                                             *J4
                             *X2                                                                            *X2
                  *EE1                                                                           *EE1
                                                                                            *FF1
                                                                                        *FF2
                                                                                *GG1
*EE3
```

Subsentence 13

```
R7/A,B,Unk,Unk/(R3((EE-EE3),R3(R1((X-X2),(Y)),R3(R1((J-J10)),(K)),(A)))),R3((GG-GG2),R12(R3((FF-FF2),R3((EE-EE3),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A))))),R3((G-G4),(A)))))
                                *X1                  *J3                                                                  *X1                *J3                       *G1
                                                  *J4                                                                                     *J4
                              *X2                                                                                     *X2
                  *EE1                                                                                          *EE1
                                                                                                          *FF1
                                                                                                      *FF2
                                                                                  *GG1
*EE4
```

Subsentence 14

```
R13(R3((FF-FF2),R3((EE-EE4),R3(R1((X-X2),(Y)),R3(R1((J-J10),(K)),(A))))),R12(R4((HH),(II)),R3((G-G4),(A))))
                                *X1                *J3                         *+(HH)(II)+    *G1
                                                *J4                          *+*HH1*II1+
                              *X2
                  *EE1
     *FF1
*FF3
```

Subsentence 15

```
R6/A,B,Unk,Unk/(R3(R1((X-X2),(S)),R3(R1((J-J10),(K)),(A))),R3((JJ),R1((M-M4),R4((KK),(G-G4)))))
                   ---------------    ---------------                              ---------------
                   *X3                *J3                                          *+(KK)(G-G4)+
                                  -----------------------                  ---------------------
                                  *J4                                      *M5
                ------------------------------------------ -------------------------------------
                *X4                                        *JJ1
------------------------------------------------------------------------------------------------
*X5
```

APPENDIX F

QUESTIONS WITH EXPLICIT ANSWERS

What are some government officials of NVN?
(?,1,1)Some(Ofl(Mod(official,government),NVN),what)
Level one response: "Government official of NVN some leader of Lao Dong party of NVN."

How is the government of NVN controlled?
(?-PL3,1,1)Control(how,Ofl(government,NVN))
Level one response: "Lao Dong party of NVN control government of NVN by means of an interlock system of parallel hierarchy."

The formal governmental structure of NVN is distinct from what?
(?,1,1)Distinct(Ofl(Mod(structure,And(form,government)),NVN), what)
Level one response: "Form government structure of NVN distinct structure of rule communist party of NVN."

What defines the formal governmental structure of NVN?
(?,1,1)Define(what,Ofl(Mod(structure,And(form,government)),NVN))
Level one response: "1960 constitution define form government structure of NVN."

What does the 1960 constitution define?
(?,1,1)Define(Mod(constitution,1960),what)
Level one response: "1960 constitution define form government structure of NVN."

Are some leaders of the Lao Dong Party of NVN government official?
(?,1,1)Some(Ofl(leader,Ofl(Mod(party,Lao Dong),NVN)),Ofl(Mod (official,government),NVN))
Level one response: "Yes"

APPENDIX G

QUESTIONS REQUIRING DEDUCTION TO ANSWER

Is the government of NVN controlled by the Lao Dong Party?
(?,1,1)Control(Mod(party,Lao Dong),Of1(government,NVN))
Level one response: "Yes"

What does the Lao Dong Party control?
(?,1,2)Control(Mod(party,Lao Dong),what)
Level one response one: "Lao Dong party of NVN control government of NVN."
Level one response two: "Lao Dong party of NVN control form machinery of government of NVN."

Is the Lao Dong Party a Communist party?
(?,1,1)Equivalent(Mod(party,Lao Dong),Mod(party,communist))
Level one response: "Yes"

Are some of NVN's government officials Communist party leaders?
(?,1,1)Some(Of1(Mod(official,government),NVN),Mod(leader,And (communist,party)))
Level one response: "Yes"

Is the formal governmental structure of NVN equivalent to the structure of the Communist party?
(?,1,1)Equivalent(Of1(Mod(structure,And(form,government)),NVN), Of1(structure,Mod(party,communist)))
Level one response: "No"

When was the governmental structure of NVN defined?
(?-P12,1,1)Define(when,Of1(Mod(structure,government),NVN))
Level one response: "1960 constitution define form government structure of NVN in 1960-P."

What does the Lao Dong Party control?
(?,1,1)Control(Mod(party,Lao Dong),what)
Level one response: "Lao Dong Party of NVN control form machinery of government of NVN."

LIOGRAPHIC DATA ET	1. Report No. UIUCDCS-R-72-500	2.	3. Recipient's Accession No.
itle and Subtitle QAS Question-Answering System			5. Report Date Feb. 1972
			6.
uthor(s) Marion Arthur Pumfrey			8. Performing Organization Rept. No.
erforming Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No.
2 ponsoring Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			13. Type of Report & Period Covered M.S. Thesis
			14.

5 upplementary Notes

6 bstracts

This thesis deals with artificial intelligence - specifically question-answering systems. A semantic net based question-answering system, QAS is developed. QAS inputs natural English sentences which have been preprocessed into simple kernel sentences and then encoded into a set of relations. QAS adds information to the memory network in a consistent manner when the information is not already represented, or answers questions based on the information represented in the memory network. QAS utilizes two question-answering modes. The first mode processes questions with explicitly represented answers. The second mode processes questions which require the use of deductive heuristics.

7. ey Words and Document Analysis. 17a. Descriptors

Question-Answering
Sentence
Linguistic
Encoding
Network

7b Identifiers/Open-Ended Terms

Natural Language Data Analysis

7c COSATI Field/Group

8. vailability Statement Release unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 81
	20. Security Class (This Page UNCLASSIFIED	22. Price